NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

COMMERCIAL OFF-THE-SHELF (COTS)/LEGACY
SYSTEMS INTEGRATION ARCHITECTURAL DESIGN AND
ANALYSIS

by

Thomas M. Nguyen

September 2000

Thesis Advisor: Mantak Shing
Second Reader: Luqi

Approved for public release; distribution is unlimited.

20001030 146

REPORT DOCUMENTATION PAGE

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 2000	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE
Commercial Off-The-Shelf (COTS)/Legacy Systems Integration Architectural Design and Analysis

5. FUNDING NUMBERS

6. AUTHOR(S)
Thomas M. Nguyen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

The nature of COTS products often falls short of achieving the unique requirements of the Department of Defense(DoD). The focus of this thesis is on the use of distributed component middleware technology within the framework of integrating COTS/Legacy system architecture. One of the main problems facing distributed computing is software component integration. There is no single, standardized framework for achieving component integration. However, technologies such as Common Object Request Broker Architecture (CORBA) and Microsoft's Component Object Model (COM) are emerging as solutions to component integration. These methodologies provide a sort of software communications bus for components, supporting platform and language independency. A case study developed within the Navy Integrated Tactical Environmental System I (NITES I) architecture was used to show the integration and communication of COTS/Legacy software components using distributed component technology. This resulted in a distributed object architecture supporting location, platform, and programming language transparencies.

14. SUBJECT TERMS
Commercial-Off-The-Shelf(COTS), Legacy System, Distributed Components, Middleware, Heterogeneous System Integration

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFI- CATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

COMMERCIAL OFF THE SHELF (COTS)/LEGACY SYSTEMS INTEGRATION
ARCHITECTURAL DESIGN AND ANALYSIS

Thomas M. Nguyen
Space and Naval Warfare Systems Center, San Diego
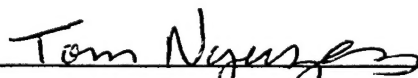B.S., University of Missouri-Rolla, 1992

Submitted in partial fulfillment of the
requirements for the degree of
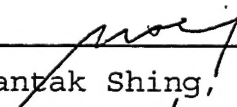
MASTER OF SCIENCE IN SOFTWARE ENGINEERING

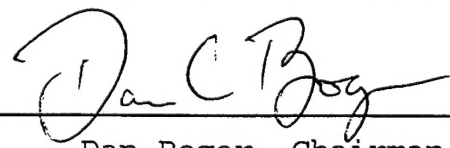from the

NAVAL POSTGRADUATE SCHOOL
September 2000

Author: _____
Thomas M. Nguyen

Approved by: _____
Mantak Shing, Thesis Advisor

_____
Luqi, Second Reader

_____
Dan Boger, Chairman
Department of Computer Science

iii

ABSTRACT

The nature of COTS products often falls short of achieving the unique requirements of the Department of Defense(DoD). The focus of this thesis is on the use of distributed component middleware technology within the framework of integrating COTS/Legacy system architecture. One of the main problems facing distributed computing is software component integration. There is no single, standardized framework for achieving component integration. However, technologies such as Common Object Request Broker Architecture (CORBA) and Microsoft's Component Object Model (COM) are emerging as solutions to component integration. These methodologies provide a sort of software communications bus for components, supporting platform and language independency. A case study developed within the Navy Integrated Tactical Environmental System I (NITES I) architecture was used to show the integration and communication of COTS/Legacy software components using distributed component technology. This resulted in a distributed object architecture supporting location, platform, and programming language transparencies.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEGEMENT

I. INTRODUCTION

A. BACKGROUND

Traditional hierarchical organizational structures are being replaced by networked organizations with emphasis on personal and organizational communication, flexibility, responsiveness, decentralized decision-making, and interoperability. Many systems developed for the Department of Defense (DoD) in the past were not designed for joint operation and interoperability. As organizations within the DoD work towards migrating to a distributed computing environment, communications services and decision support systems are becoming important aspects of the information technology infrastructure. The objective of this thesis is to identify and analyze the distributed software components and object middleware technology, which can be used to integrate computing systems and applications together in a distributed computing architecture.

B. SCOPE AND ORGANIZATION

The scope of this thesis is to develop and analyze distributed software component technology in researching how to achieve interoperability between software systems. The overall goal in respect to the DoD is interoperability

1

throughout the Department of Defense as well as with our international allies. This thesis is the result of a joint group effort in researching and designing an architectural framework, which integrates Commercial-Off-The-Shelf (COTS) component and application technology with aging legacy systems. It focuses on the distributed components and middleware technology within the architectural framework. Topics such as security, wrappers, and overall framework are outside the scope of this thesis and they are covered elsewhere in conjunction to this report [Gee][Tran].

We will start with an overview of the Distributed Computing Architecture, and then discuss component Object Request Broker (ORB) technology and how to achieve interoperability among objects. Chapter IV identifies the Communications Requirements and Specifications in reference to the Legacy Systems Integration's Architectural Framework Analysis and Design. Chapter V describes how we applied distributed computing concepts covered in this research to solve problems associated with integrating a current Department of Defense (DoD) legacy integration project called the Navy Integrated Tactical Environmental Support System I (NITES I).

II. DISTRIBUTED COMPUTING ARCHITECTURE OVERVIEW

A.   CLIENT/SERVER COMPUTING

The basic concept for communication between a client
and server consists of a request from a client to a server
and a response from the server back to the client. One way
to implement this is by utilizing Interprocess
Communications (IPC) such as messaging and Remote Procedure
Call (RPC). The client/server applications provide a
method to transparently and most efficiently access the
information the user needs from a network of resources.
Various application components or tasks are distributed
between client and server platforms, which cooperate to
perform the desired application functions. Communications
between the client and server can be viewed as interactions
between interconnected components of the client/server
architecture. Each component provides an interface through
which it communicates with other components. Two distinct
classes of components can be defined as:

- Process components: Typically, these are software
  components that actively perform some functions.

- Resource components: These provide the services
  requested by process components.

Client/server interactions can be separated from the interprocess communications and network protocols by utilizing a set of common interfaces and run-time facilities, which will allow client/server interaction to be developed and performed totally transparent to the programmers and end users. Since these common interfaces and run-time facilities are architecturally layered between clients and servers, they are widely known as middleware. Designing an efficient client/server application can be challenging, the goal is to evenly distribute execution of tasks between processors while making optimal use of available resources.

1. Database Protocols

The X/Open Call Level Interface (CLI) specification provides an interface to Relational Database Management Systems (RDBMS) using Structured Query Language (SQL). Microsoft's Open Database Connectivity (ODBC) Application Programming Interface (API) is the best known implementation of the CLI standard. Sun Microsystems' Java Database Connectivity (JDBC) API is a new implementation of the CLI standard specifically for Java applications.

The CLI architecture are perhaps the most commonly envisioned usage of client/server computing, in most cases,

4

allowing applications written using the standard to operate independent of the database to which they are connected. The drawback is that it does not provide access to some of the more advanced features that differentiate RDBMS products.

The API presented by the specification varies in appearance from a thinly-veiled messaging interface to an RPC interface. The message-like components of the interface expose a hybrid synchronous/asynchronous mode of operation wherein initial results are returned synchronously while processing may continue asynchronously at the server. This allows the client to continue processing as soon as the server is able to provide an initial set of results; further results are queued by the server and returned to the client as they are requested. The RPC components are used for control purposes and operate synchronously.

B. MIDDLEWARE

Although middleware comes in many different forms, its basic function is to enable interprocess communication. Conceptually, it is the glue that holds together the disparate systems in a distributed computing environment. Architecturally, middleware functions as a layer of the

client/server architecture that resides between the client

and the server. In addition, it supports multiple

communication and data access protocols and interfaces, and

enables run-time client/server interactions. Middleware

integrates application programs and other software

components in a distributed environment which can be

characterized by:

- Distribution of processing among multiple systems
- Interactions between dissimilar systems
- Ability to share resources between individual
  interconnected systems
- Multiple specialized and heterogeneous nodes and
  networks   [Berson]

As a software layer, middleware designed to be a

common software component that sits between clients and

servers on top of the communication protocols and frees

client applications from the need to know low-level

communication protocols. It is not designed to replace

communication protocols. In order to integrate

applications in a distributed environment and to take

advantage of the functionality provided by communication

networks, middleware provides an abstraction layer that, at

a minimum, should enable:

- Node, service, and data location transparency
- Seamless interactions between application components
  via a set of common APIs
- Scalability and Extensibility

6

- Reliability and Availability
- Vendor, platform, operating system, and networking protocol independence. [Berson]

1. Database Middleware

Most two-tier client/server applications are built using proprietary database middleware supplied by a DBMS vendor. The middleware ships SQL requests over the network to a relational DMBS and returns data to the client application. The database gateway represents another example of database middleware. This database system supports remote database access to more than one database engine using an ODBC API.

2. Application Middleware

Application middleware differs from database middleware in a significant way. Database middleware lets user-written components talk to supplied database engines or web browsers. The developer has very little design flexibility, because the software vendor has already defined most of the rules for the communication. Application middleware, in contrast, is more like a general-purpose programming language. It allows two user-written components to communicate in any way that suits the application designer and developer. The choice of

communication style is a key application design decision,

particularly with regards to the question of whether to

implement synchronous or asynchronous connections.

3. Message-Oriented Middleware

Message-oriented middleware hides the network from

applications and programmers. In addition, it supports

asynchronous communications. Message-queuing software has

traditionally been used in transaction processing. For

this purpose, message-queuing software has transactional

functionality such as database commits and roll backs,

where the stored messages are persistent. The difference

between traditional and modern client/server message-

queuing software is that traditional message-queuing

software had proprietary interfaces and was tied to

proprietary network, operating systems, and hardware. In

contrast, modern message-queuing software is network,

operating-system, and hardware independent. In addition,

the interfaces are published and available to other vendors

for use with their applications.

C. DISTRIBUTED-TRANSACTION PROCESSING

Distibuted-transaction processing (DTP) allows an

organization to distribute transactions and transaction

data across multiple, geographically dispersed, autonomous nodes in such a way that a node anywhere in a network can initiate or process a transaction on any one or more network nodes. The location of the transaction manager that manages a transaction, and the data required for the transaction, are transparent to users and applications. DTP requires specialized standardized transaction processing interfaces, protocols, and formats to allow the different parts of the DTP system to communicate with one another. DTP in the form of Transaction Processing Manager/Monitor (TPM) will be discussed in more detail in the next section.

THIS PAGE INTENTIONALLY LEFT BLANK

III. DISTRIBUTED OBJECTS COMPUTING

A. COMPONENTS AND DISTRIBUTED OBJECTS

> Distributed object technology is extremely well-
> suited for creating flexible client/server
> systems because the business logic are
> encapsulated within objects, allowing them to be
> located anywhere within a distributed system.
> [Orfali]

Orfali contends that objects and components are
revolutionizing the way we assemble our client systems.
Because distributed objects such as Common Object Request
Broker Architecture (CORBA), Java, and Component Object
Model (COM) provide information about themselves,
applications and visual tools are able to know the object's
interfaces, events and property instantaneously. Thus
distributed objects are able to:

1. Allow granular components to interoperate across
   networks.
2. Run on different platforms.
3. Coexist with legacy applications through object
   wrappers.
4. Manage themselves and the resources they control.

Like well-designed procedural APIs, implementation
details are hidden from the user of the object. Unlike

11

traditional APIs, however, object architectures limit

access to the invocation of methods defined for the object.

In addition, methods are invoked on the objects indirectly,

via references to the objects, eliminating the need for

local instances of the objects. This near-complete

implementation hiding allows distributed object

architectures to support location, platform, and

programming language transparency. These transparencies

come at a price (e.g. cost of extra overhead).

One of the main problems facing distributed computing

is software component integration. There is no single,

commercially available, widely recognized, and standardized

approach and framework for achieving this integration. Two

infrastructures (also referred to as Object Request Broker

(ORB) technology) are competing to provide a communications

software bus for components: CORBA/JavaBeans and COM.

B.   COMPONENT OBJECT MODEL/DISTRIBUTED COM (COM/DCOM)

Microsoft's solution to distributed middleware

technology is the Component Object Model (COM). It

originated as a document structuring technology called

Object Linking and Embedding (OLE), and was later evolved

into an object-oriented architecture. COM, also known as

ActiveX, is referred to as an object-based programming

12

model and a set of related system services designed to provide software interoperability. The primary goal of COM is to provide a means for client objects to make use of server objects, regardless of programming languages or who developed the objects. In order to achieve interoperability, COM defines a binary standard, which specifies how an object is laid out in memory. A COM object, like other objects, is an instantiation of a particular defining class at run-time. The difference is that instead of using a man-readable name to identify itself, a COM object uses a unique Class Identifier (CLSID) to uniquely identify the object classes. CLSIDs are part of a special group of identifiers called Globally Unique Identifiers (GUIDs). GUIDs are 128-bit values that are statistically guaranteed to be unique across time and space.

A COM object is defined in terms of the individual interfaces that it supports. Interfaces are essential to COM programming because they are the only way to interact with a COM object. An interface is identified by a unique identifier, usually called an Interface Identifier (IID). A COM client obtains a pointer to a particular interface to gain access to the functions defined as part of that particular interface. To support interface navigation,

13

every interface uses a special *QueryInterface* and *IUnknown* interface function call. *QueryInterface* contains two parameters, one to specify the desired interface's IID, and the other to receive the interface pointer. The defining factors of an interface are:

- The number of supported functions
- The function prototypes of each supported function
- The order in which the function prototypes are listed

Because the interfaces are the only way to access a COM object, changing any of these factors effectively changes the interface. Architecturally, an interface is a pointer to a virtual function table (VTBL). The VTBL contains pointers to functions that provide that actual implementation defined by the interface. Figure 1 shows that an interface is actually a pointer to a VTBL of function pointers. [Redmond]

```
COM Object

                                                         function 1
                                                         {
                                                         ...
                                                         }
COM Client        ISomeInterface    Vitual Function
                                    Table (VTBL)
                                                         function 2
pISomeInterface  →   pVTBL   →      &function 1          {
                                                         ...
                                    &function 2          }
                                    &function 3
                                                         function 3
                                                         {
                                                         ...
                                                         }
```
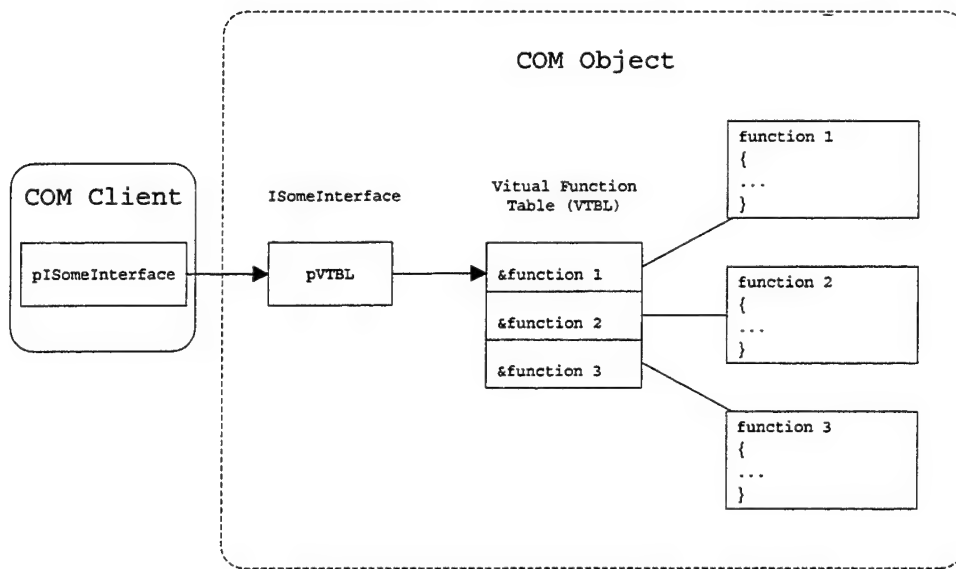
Figure 1: COM Interface to Function Pointers

1.   Distributed Component Object Model (DCOM)

DCOM expands Microsoft's COM architecture to support

communication of distributed objects in a networking

environment. Microsoft based its DCOM protocol on Remote

Procedure Call (RPC) standards developed for the

Distributed Computing Environment (DCE) standards. Called

Object RPC (ORPC), it can use Transmission Control Protocol

(TCP) for guaranteed connectivity or User Datagram Protocol

(UDP) for connectionless transfer. [Carr]

DCOM enables clients to transparently communicate with

server objects, regardless of where these objects are

running. Clients are not aware of where the server objects

15

are located. To a client, all server objects are accessed
through interface pointers.

2.   COM Server Access

A COM server provides the necessary structure around
an object to make it available to clients. It is usually a
block of code in the form of a Dynamically Linked Library
(DLL) or an executable (EXE). COM servers implemented as
DLLs, also known as in-process servers, loads a copy of the
server code directly into each client application's own
address space. This is because DLLs do not maintain their
own address space, so the server creates an instance of a
class using the IclassFactory interface. A copy of all
global resources is created on the client. COM servers
created as stand-alone EXEs maintain their own address
space and are also known as out-of-process servers. Out-
of-process servers that execute on the same machine with a
client are referred to as Local Servers. A client process
uses DCOM's Lightweight RPC (LRPC) to access a local
server. COM Servers running on a separate machine from its
client are called remote servers. Figure 2 shows COM's
transparent LRPC and RPC mechanism. [Orfali]
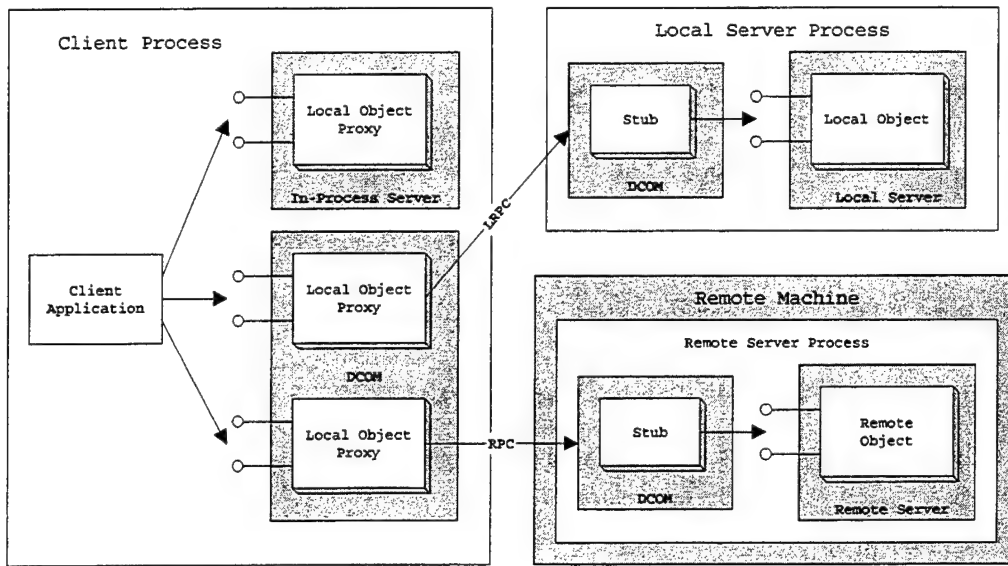
Figure 2: COM's Transparent LPC and RPC Mechanism

As mentioned earlier, DCOM allows clients to access

server objects transparently through the use of interface

pointers. Any call to an interface function must first go

to an in-process piece of code referred to as a proxy.

Clients of local out-of-process objects communicate with an

in-process proxy, which communicates with a stub loaded

into the address space of the object via LRPCs. Clients of

remote objects communicate with an in-process proxy, which

communicates with a remote stub via RPCs. The DCOM proxy

and stub mechanism uses similar concepts to that of CORBA's

implementation of static stubs on the client side and

interface skeletons on the server side.

17

In a report sponsored by the Defense Information
System Agency (DISA), the following were their
recomendations for using COM/DCOM:

- **Consider DCOM mostly for experimental use aimed at pure Windows 2000 networks.** Any impact that DCOM has on the middleware market is most likely to appear first in pure Windows 2000 networks, which will provide stronger DCOM support and also make more extensive use of DCOM within the operating system itself. DCOM will also benefit greatly when Active Directory technology in Windows 2000 becomes widely available. Using DCOM thus may be appropriate for projects that will run only Windows 2000 when it becomes available.

- **Avoid DCOM as the only middleware product for heterogeneous networks.** Current levels of support for DCOM on non-Windows operating systems do not easily justify the use of DCOM for heterogeneous networks. The current (early 1998) relative immaturity of the distributed communication features of DCOM also works against it for use in complex network environments that require multi-vendor support and a high level of adaptability to unique circumstances. (Note: This is a rapidly-changing area, and new products and changes to DCOM may make heterogeneous use of DCOM or its future incarnations such as COM+ easier in the future – e.g., sometime in 1999 or 2000. At present, however, both pure CORBA and bridging between middleware products appear to be more viable middleware approaches for heterogeneous networks that require the use of both Unix and Windows.)

- **Avoid using DCOM to integrate legacy systems (except for DCE legacy systems).** As of early 1998, DCOM does not have the flexibility or range of platform implementations needed to make it appropriate for integrating legacy (e.g., Cobol, Ada, or C++) software into new network applications. In contrast, CORBA is a much better choice for such integration activities because of its broad platform and vendor support, and because of its cleaner and more understandable object model. One important exception to this general rule

18

is that when the legacy system already uses DCE and the new portion of the network application consists of Windows-based PCs, DCOM may provide easier integration of the DCE components into the new Windows-based platforms than would CORBA in the same situation. This is possible because DCOM uses a communication protocol that is very close to that of DCE, so that a minimum of new software development should be needed to bridge between the two. Even in this case the tradeoffs of using CORBA versus DCOM for the integration should be carefully considered, however, especially if the new components of the system include both Unix and Windows operating systems.

- **Consider bridging in heterogeneous networks that require the use of DCOM.** When DCOM is required for the NT portions of a heterogeneous network, the possibility of using a middleware bridge should be considered strongly. The alternative of attempting to use DCOM across a heterogeneous network is much less attractive and in many cases may simply not be feasible. Also, the trend of CORBA vendors towards providing good bridges to COM has accelerated, with many new products likely in 1998. [DISA]

C.   COMMON OBJECT REQUEST BROKER ARCHITECTURE (CORBA)

The Common Object Request Broker Architecture

(CORBA) is a distributed object architectural

framework developed by the Object Management Group

(OMG) in 1989. CORBA allows applications to

communicate with one another no matter where they are

located or who designs them. The OMG approach to

distributed computing are:

> *To adopt interface and protocol specifications that define an object management architecture supporting interoperable applications based on distributed interoperable objects.*

19

The specifications are to be based on existing
technology that can be demonstrated to satisfy
OMG's Technical Objectives. [Bloor]

CORBA objects are components that can be anywhere on a

network, that clients can access via method invocations.

Both the language and location used to create server

objects are totally transparent to the clients. It can be

in the same process or on a different machine on a network.

1.   OMA-Object Management Architecture

The OMG has developed a conceptual model, know as the

core object model, and a reference architecture, called the

Object Management Architecture (OMA) upon which

applications can be constructed. [Yang]

The OMA consists of four components: Object Request

Broker (ORB), Object Services (OS), Common Facilities (CF),

and Applications Objects (AO) as shown in Figure 3.
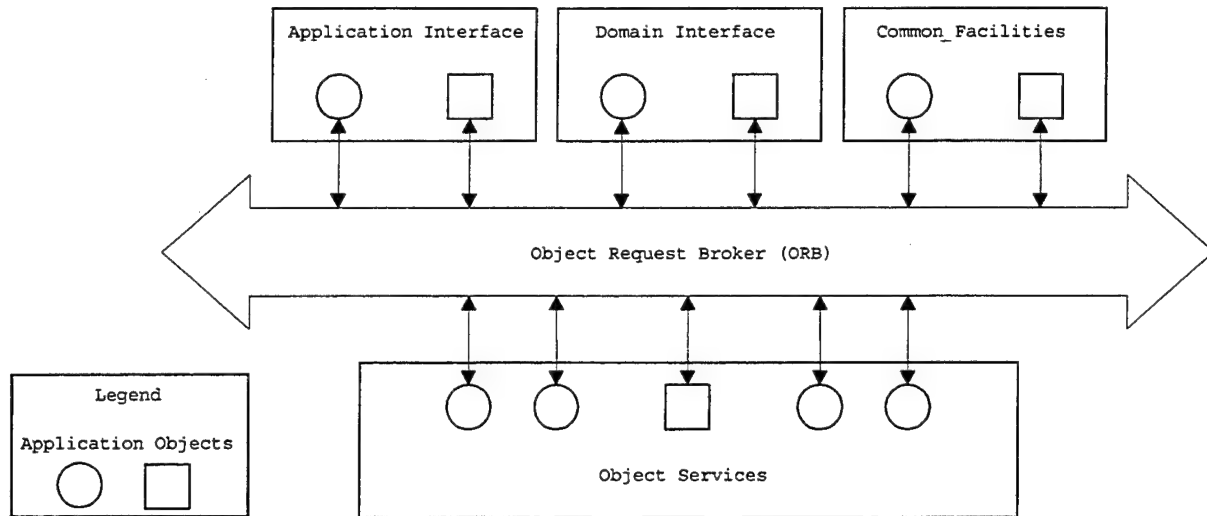
Figure 3: Object Management Architecture (OMA)

2.   Object Request Broker (ORB)

ORB is the fundamental part of the Common Object Request Broker Architecture. The primary responsibility of the ORB is to resolve requests for object references, enabling application components to establish connectivity with each other. When an application component wants to use the services of another application component, it sends a request call to the ORB. The ORB interprets the call and tries to find references to an object that can perform the service. The ORB then passes the parameters along to the object where the component can call the methods of the application and return the result.

3. Object Services (OS)

Object services provide fundamental object interfaces necessary for building object-oriented distributed applications. The operations provided by Object Services are specified in the Interface Definition Language (IDL). These are domain-independent interfaces that are used by many distributed object programs. For example a service providing for the discovery of other available services is almost always necessary regardless of the application domain. Two examples of Object Services that fulfill this role are:

- The Naming Service - which allows clients to find objects based on names.
- The Trading Service - which allows clients to find objects based on their properties.

4. Common Facilities (CF)

Common Facilities provide standardized interfaces to common application services. Like Object Service interfaces, these interfaces are also horizontally-oriented, but unlike Object Services they are oriented towards end-user applications. An example of a common facility is the Distributed Document Component Facility (DDCF), a compound document Common Facility based on

22

OpenDoc. DDCF allows for the presentation and interchange of objects based on a document model, for example, facilitating the linking of a spreadsheet object into a report document.

5. Application Object (AO)

Application Objects are interfaces developed specifically for a given application. AOs are not standardized because they are application specific, and because the OMG does not develop applications, only specifications.

6. Interface Definition Language (IDL)

Another fundamental part of the CORBA architecture is the Interface Definition Language (IDL). IDL is a standard language used to specify the interfaces used between CORBA objects. IDL specification is responsible for ensuring that data is properly exchanged between dissimilar languages. Because interfaces described in IDL can be mapped to any programming language, CORBA applications and components are independent of the languages used to implement them. For example, a client written in C++ can communicate with a server written in Java, which in turn

can communicate with another server written in COBOL, and so on.

Because IDL is not an implementation language, you can not write applications in IDL. The main purpose for IDL is to define interfaces, implementating the interfaces is performed using some other language.

7. CORBA Communications Model

When a component of an application wants to access a CORBA object, it first obtains an Interoperable Object Reference (IOR) for that object. Using the IOR, the component (called a client of that object) can then invoke methods on the object (called the server in this instance).

In CORBA, a client is simply any application that uses the services of a CORBA object; that is, an application that invokes a method or methods on other objects. Likewise, a server is an application that creates CORBA objects and makes the services provided by those objects available to other applications. As previously mentioned, CORBA ORBs usually communicate using the Internet Inter-ORB Protocol (IIOP). Other protocols for inter-ORB communication exist, but IIOP is fast becoming the most popular, first of all because it is the standard, and second because of the popularity of Transmission Control

24

Protocol/Internet Protocol (TCP/IP) (the networking protocols used by the Internet), a layer that IIOP sits on top of. CORBA is independent of networking protocols, however, and could (at least theoretically) run over any type of network protocols.

8. CORBA Object Model

In CORBA, all communication between objects is done through object references, these are known as Interoperable Object References(IORs). Visibility to objects is provided only through passing references to those objects, this means that remote objects in CORBA remain remote, there is currently no way for an object to move or copy itself to another location. Another aspect of the CORBA object model is the Basic Object Adapter (BOA), a BOA provides the common services available to all CORBA objects. In CORBA, a component can act as both a client and as a server. A component is considered a server if it contains CORBA objects whose services are accessible to other objects. Likewise, a component is considered a client if it accesses services from some other CORBA object. [Rosenberger]

9.   Stubs and Skeletons

A client stub is a small piece of code that allows a client component to access a server component. This piece of code is compiled along with the client portion of the application. Similarly, server skeletons are pieces of code that is provided when you implement a server. The client stubs and server skeletons are generated when you compile IDL interface definitions.

10.   CORBA Interface Architecture

The CORBA specification defines an architecture of interfaces consisting of three specific components: client-side interface, object implementation side interfaces, and ORB Core, as shown in Figure 4.
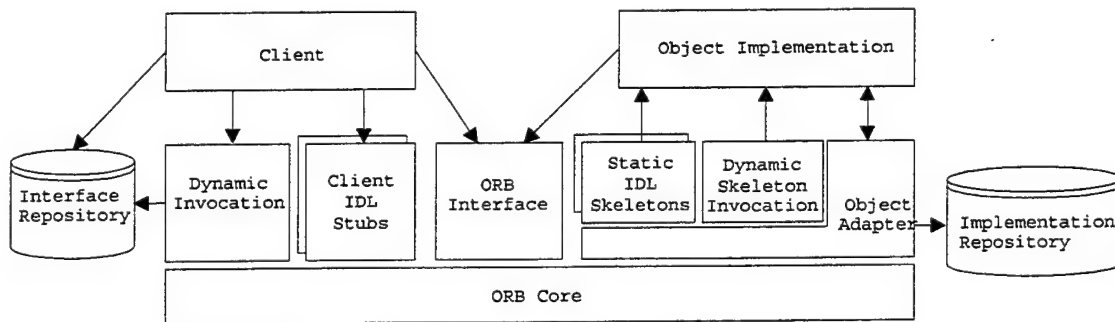
Figure 4: The structure of CORBA interfaces

11. Client Interface

Looking at the structure of the CORBA interface, the Client Interface may either use the Dynamic Invocation Interface (DII), or call a statically defined IDL Stub to make a request to the object.

12. IDL Stubs

An IDL stub includes functions generated from IDL interface definitions and linked into the client program. This is the static invocation interface, representing a language mapping between the client language and the ORB implementation, providing the static interfaces to object services. The stub contains code that encodes and decodes the operation and its parameters into message formats that can be sent to the server. This process is called marshaling.

13. Dynamic Invocation Interface (DII)

Dynamic Invocation Interface (DII) is generated at run time. It is used when the object interface is not known at compile time. Using DII, an object is accessed by a call to the ORB or by a series of calls to the ORB in which the object, method, and parameters are specified. The client

27

has the responsibility of specifying the types of parameters and expected results.

14. ORB Interface

The ORB Interface is the interface that goes directly to the ORB which is the same for all ORBs and does not depend on the object's interface or object adapter. Because most of the functionality of the ORB is provided through the object adapter, stubs, skeleton, or dynamic invocation, there are only a few operations that are common across all objects. These operations are useful to both clients and implementetation of objects.

15. IDL Skeleton

The IDL Skeleton, also referred as Server IDL Stub, is created using an IDL compiler and resides on the server side.

16. Dynamic Skeleton Interface (DSI)

DSI provides a run time binding mechanism for servers to deliver request from an ORB to an object implementation that does not have compile-time knowledge of the type of the object. The dynamic skeleton inspects the parameters of an incoming request to determine a target object and

method.   When a client on one ORB calls a server on another ORB, the DSI transmits the request to the target ORB, and then the bridge uses the dynamic invocation interface to invoke the target object on that ORB.   The DSI can receive either dynamic or static invocation from clients.

17.   Generic InterORB Protocol (GIOP)

The Generic InterORB Protocol (GIOP) is the base protocol (in terms of messages) for official interORB communication.   This can include vendor-specific proprietary communication, so GIOP must be able to map to any connection-oriented medium.   The OMG specifies three parts to GIOP:

- The Common Data Representation (CDR)
- The various GIOP message formats
- The message transport assumptions

The CDR is essentially a low-level transfer syntax that maps between OMG IDL types and low-level raw data types for use between network agents and processes.

18.   Object Adapter (OA)

The Object Adapter sits on top of the ORB Core communication services, accepting requests for service to the requested objects.   The object implementation accesses

29

most ORB services and the ORB Core through the Object
Adapter. It provides the run-time environment for
instantiating server objects, passing requests to them, and
assigning object references to them.

An object adapter is the primary way that an object
implementation accesses services provided by the ORB.
Services provided by the ORB through an Object Adapter
often include: generation and interpretation of object
references, method invocation, security of interactions,
object and implementation activation and deactivation,
mapping object references to implementations, and
registration of implementations.

19. ORB Core

The ORB Core provides a mechanism for a client to
transparently communicate with objects without the client
having to know the details of the method invocations. This
makes the client requests appear to be local procedural
calls.

20. Repositories

The interface repository provides another way to
specify the interfaces to objects. Interfaces can be added
to an Interface Repository service which defines operations

for run time retrieval of information from the repository.
Using an interface repository, a client should be able to
locate an object unknown at compile time, enquire about its
interface, and then build a request to be forwarded through
the ORB. The interface repository allow you to obtain the
interface and modify the descriptions of all the component
interfaces during initialization. In addition, using the
information in the Interface Repository, it is possible for
a program to come across an object which is unknown at
compiled time, but still be able to dynamically determine
what operations are valid on the object and make an
invocation on it.

The implementation repository contains information
that allows the ORB to locate and activate implementations
of objects. Although most of the information in the
Implementation Repository is specific to an ORB or server
object, the Implementation Repository is where all the
information about the classes a server support is stored.
In CORBA version 1.1, interoperable object applications was
not totally achieved because the OMG left the
implementation of the ORB core to vendor preferences. This
only resulted in some level of component portability, but
not interoperability. CORBA 2.0 fixed the interoperability
problem by specifying a mandatory Internet Inter-ORB

Protocol (IIOP). IIOP specifies how ORBs communicate over TCP/IP, using the internet as the backbone through which other ORBs can bridge to.

21. Performance

There are three main factors that reduces throughput when using CORBA in comparison to TCP sockets. The first factor is increased message overhead caused by Generic InterORB Protocol (GIOP) and Common Data Representation (CDR). CDR adds padding into data structures to maintain alignment. GIOP header can add to the message size. The header begins with a 12-byte field providing version and message type information.

The second cause of CORBA overhead is marshaling. For data structures, the sending ORB must collect the data from different location in memory and copy it into a transmit buffer. If a parameter is declared with the generic type in IDL, the sending ORB must include specific type data whenever the parameter is marshaled.

The third cause is reduced throughput via dispatching. The sending application calls the client-side stub, which calls the ORB to perform the remote operation. On the receiver, the ORB calls an adapter interface, which performs an up call to the server. Overhead can be

significant if the stubs are implemented inefficiently, or cause unnecessary copying.

The general rule is; for distributed applications where bandwidth and latency requirements are below what is available at the transport layer, CORBA is the ideal solution. However, for applications requiring high performance, developers should carefully consider the tradeoffs before committing to CORBA development.

The following are recommendations from DISA for using CORBA to build applications:

- **Use CORBA**. The *de facto* position of CORBA as the most widely distrbuted and used middleware product for Internet-connected PCs makes it an excellent choice for low end Windows platforms. Furthermore, its broad availability and support on other platforms such as Unix makes it useful for integrating legacy software is further enhanced by its clear, well-defined, and internationally standardized interface description language for specifying the interfaces to software components. The maturity of the object-oriented features of CORBA also make it well-suited to the current trend towards more dynamic distributed software that whose relationship to the underlying network can change in real time.

- **Use only one CORBA vendor unless interoperability can be verified**. The greatest current weakness of CORBA is the slow pace of its efforts to make CORBA products from different vendors interoperate with each other. There has been significant progress in this area in the last couple of years, but at present the safest strategy for using CORBA is still to pick a single vendor and use that vendor consistenly for a given application.

- **Use "gang of three" (CORBA/Internet/Java) CORBA products whenever possible**. At present, the most

33

promising overall path for broad integration of applications using the Internet appears to-be joint use of CORBA (especially IIOP), Internet technologies, and Java. CORBA provides integration of legacy systems, broad platform interoperability, object-oriented interfaces, and a well-defined path (the ORB) for implementing various forms of network transparency. This support is likely to become increasingly important as part of an overall industry thrust to make distributed applications more scalable, robust, and portable. The Internet provides a robust universal network for hosting distributed applications, and Java provides a dynamic programming "glue" that can be used to develop new interfaces into older legacy systems more rapidly and more effectively. While all CORBA 2.0 and 2.1 products are required to support IIOP, the way in which IIOP is supported can vary significantly from vendor to vendor. The best implementations make good use of features that increase efficiency and reduce needless overhead for operations such as communication with Java objects.

- **For networks that include NT, use CORBA with good COM bridges.** DCOM will be an important force in the upcoming release of Windows NT 5.0. However, for now (early 1998) an approach that relies on CORBA for the network side of distributed applications and CORBA-TO-COM bridges for the Windows NT and Windows 95 side is more likely to product robust, reliable distributed applications. Support for CORBA-to-COM bridges should increase in 1998, as demonstrated by the early 1998 release of products such as the IONA OrbixCOMnet Desktop. [DISA]

D.   JAVABEANS

JavaBeans is the most recent of the three

specifications of component architecture for building

applications as reusable components. The JavaBeans

architecture is where Java extends from a simple

development language to component model technology like

CORBA and COM. JavaBeans allows Java developers to create replaceable code within applications, and applications can be componentized so that the individual elements can be reused within other application systems without re-coding. These components are designed to work within any Java application builder tool and execute within anything that has a Java Virtual Machine (JVM), including web browsers. Beans can be manipulated and customized through their property tables and customization methods. In addition, multiple beans can be combined to create more sophisticated applets, applications, or other JavaBeans.

1. Enterprise JavaBeans (EJB)

Extending behind the client-side based JavaBeans model, Enterprise JavaBeans is a Java-based cross-platform component architecture for the development and deployment of multi-tier, distributed, scalable, server-based, object-oriented Java applications. EJB simplifies writing business applications as components by providing a set of automatic services to support scalable transactional application server components.

2. Java RMI

Java Remote Method Invocation (RMI) is a set of packages included in the JDK which allow for Java-to-Java communication between distributed application components. The following are Java RMI goals:

- Support remote invocation on distributed Java objects.
- Support callbacks from servers to applets.
- Distributed object model integration into Java.
- To differentiate between the distributed object model and local Java object model.
- Simplify writing reliable distributed applications.
- Preserve the security attributes in the Java runtime environment.

Java/RMI uses a protocol called the Java Remote Method Protocol (JRMP), which relies on Java Object Serialization, allowing objects to be distributed as a stream. Each Java/RMI Server object defines an interface which can be used to access the server object outside of the current Java Virtual Machine (JVM) and on the other machine's JVM. The interface reveals the services that is offered by the server object.

36

E.   EXTENSIBLE MARKUP LANGUAGE (XML)

Another increasing popular trend is web-based applications. The Extensible Markup Language (XML), an extension to the Hypertext Markup Language (HTML), has become a powerful tool for web development. Both HTML and XML may appear the same in a browser, but the XML data is "smart" data. HTML tells how the data should look to the browser, but XML tells the browser what it means. With XML, the developer can create their own tags to describe what they want the data to mean. This essentially makes it a smart document. XML is used to put data in a format that is computer-readable code so that we can use the computer to process or store the data.

XML documents are composed of markup and content. There are six kinds of markup that can occur in an XML document: elements, entity references, comments, processing instructions, marked sections, and document type declarations. The following sections introduce each of these markup concepts.

1.   Elements

Elements are the most common form of markup. Delimited by angle brackets, most elements identify the nature of the content they surround. Some elements may be empty, as seen above, in which case they have no content. If an element

37

is not empty, it begins with a start-tag, <element>, and
ends with an end-tag, </element>.

2. Attributes

Attributes are name-value pairs that occur inside
start-tags after the element name. For example,
<div class="preface"> is a div element with the attribute
class having the value preface. In XML, all attribute
values must be quoted.

3. Entity References

In order to introduce markup into a document, some
characters have been reserved to identify the start of
markup. The left angle bracket, <, for instance,
identifies the beginning of an element start- or end-tag.
In order to insert these characters into your document as
content, there must be an alternative way to represent
them. In XML, entities are used to represent these special
characters. Entities are also used to refer to often
repeated or varying text and to include the content of
external files. Every entity must have a unique name. In
order to use an entity, you simply reference it by name.
Entity references begin with the ampersand and end with a
semicolon. For example, the lt entity inserts a literal <

into a document. So the string <element> can be represented in an XML document as < element>.

A special form of entity reference, called a character reference can be used to insert arbitrary Unicode characters into your document. This is a mechanism for inserting characters that cannot be typed directly on your keyboard. Character references take one of two forms: decimal references, ℞, and hexadecimal references, ℞. Both of these refer to character number U+211E from Unicode.

4.   Comments

Comments begin with <!-- and end with -->. Comments can contain any data except the literal string --. Comments can be placed between markup anywhere in a document. Comments are not part of the textual content of an XML document. An XML processor is not required to pass them along to an application.

5.   Processing Instructions

Processing instructions (PIs) are an escape hatch to provide information to an application. Like comments, they are not textually part of the XML document, but the XML processor is required to pass them to an application.

Processing instructions have the form: <?name pidata?>.
The name, called the PI target, identifies the PI to the
application. Applications should process only the targets
they recognize and ignore all other PIs. Any data that
follows the PI target is optional, it is for the
application that recognizes the target. The names used in
PIs may be declared as notations in order to formally
identify them. PI names beginning with XML are reserved
for XML standardization.

6. CDATA Sections

In a document, a CDATA section instructs the parser to
ignore most markup characters. Consider a source code
listing in an XML document. It might contain characters
that the XML parser would ordinarily recognize as markup (<
and &, for example). In order to prevent this, a CDATA
section can be used.

```
<![CDATA[
*p = &q;
b = (i <= 3);
]]>
```

Between the start of the section, <![CDATA[and the
end of the section,]]>, all character data is passed
directly to the application, without interpretation.

Elements, entity references, comments, and processing instructions are all unrecognized and the characters that comprise them are passed literally to the application. The only string that cannot occur in a CDATA section is]]>.

7.   Document Type Declarations

A document type definition (DTD) is a series of deinitions for element types, attributes, entities and notations. It declares which of these are legal within the document and in what places they are legal. A large percentage of the XML specification deals with various sorts of declarations that are allowed in XML. One of the greatest strengths of XML is that it allows you to create your own tag names. But for any given application, it is probably not meaningful for tags to occur in a completely arbitrary order. So, if the document is to have meaning, there must be some constraint on the sequence and nesting of tags. Declarations are where these constraints can be expressed.

Additionally, declarations allow a document to communicate meta-information to the parser about its content. Meta-information includes the allowed sequence and nesting of tags, attribute values and their types and

defaults, the names of external files that may be referenced and whether or not they contain XML, the formats of some external (non-XML) data that may be referenced, and the entities that may be encountered.

8.   Benefits of XML

The following are benefits of XML:

- **Simplicity** - Information coded in XML is easy to read and understand, plus it can be processed easily by computers.

- **Openness** - XML is a W3C standard, endorsed by software industry market leaders.

- **Extensibility** - There is no fixed set of tags. New tags can be created as they are needed.

- **Self-description** - In traditional databases, data records require schemas set up by the database administrator. XML documents can be stored without such definitions, because they contain meta data in the form of tags and attributes. XML Provides a basis for author identification and versioning at the element level - Any XML tag can possess an unlimited number of attributes such as author or version.

- **Contains machine readable context information** - Tags, attributes and element structure provide context information that can be used to interpret the meaning of content, opening up new possibilities for highly efficient search engines, intelligent data mining, agents, etc. This is a major advantage over HTML or plain text, where context information is difficult or impossible to evaluate.

- **Separates content from presentation** - XML tags describe meaning not presentation. The look and feel of an XML document can be controlled by XSL style sheets, allowing the look of a document (or of a complete Web site) to be changed without touching the content of the document. Multiple views or presentations of the same content are easily rendered.

- **Supports multilingual documents and Unicode** - This is important for the internationalization of applications.

- **Facilitates the comparison and aggregation of data** - The tree structure of XML documents allows documents to be compared and aggregated efficiently element by element.

43

- **Can embed multiple data types** - XML documents can contain many data types - from multimedia data (image, sound, video) to active components (Java applets, ActiveX).

- **Can embed existing data** - Mapping existing data structures like file systems or relational databases to XML is simple. XML supports multiple data formats and can cover all existing data structures.

- **Provides a 'one-server view' for distributed data** - XML documents can consist of nested elements that are distributed over multiple remote servers. XML is currently the most sophisticated format for distributed data - the World Wide Web can be seen as one huge XML database.

- **Rapid adoption by industry** - Software AG, IBM, Sun, Microsoft, Netscape, DataChannel, SAP and many others have already announced support for XML. Microsoft will use XML as the exchange format for its Office product line, while both Microsoft's and Netscape's Web browsers support XML. SAP has announced support of XML through the SAP Business Connector with R/3. Software AG supports XML in its Bolero and Natural product lines and provides Tamino, a native XML database. [Software AG]

44

F. TRANSACTION PROCESSING MONITORS (TPM)

Transaction Processing Monitors (TPM) used as
middleware solutions can support transaction routing,
execution of remote functions, and transparent access to
remote data, transaction integrity, manageability, and
recoverability. TP monitors first appeared on mainframes
to provide run-time environments that could support large-
scale On-Line-Transaction-Processing (OLTP) applications
such as airline reservations, banking and stock-brokerage
systems. Since then, TPM have been combined with ORBs into
Object Transaction Monitors (OTMs) to better manage CORBA
and DCOM objects. Most TPMs now provide C++ class
libraries to access their services. TPMs like Encina,
CICS, Tuxedo, and Top End, allow CORBA clients to call
their services using CORBA IDL interfaces and IIOP. OTM
are capable of managing millions of objects, coordinating
their interactions across the network. Figure 5
illustrates the use of OTMs to manage various ORB
components. Instead of coordinating procedural services,
an OTM manages server-side components such as CORBA Beans
and Enterprise JavaBeans. Microsoft's version of an OTM is
the Microsoft Transaction Server (MTS), which is an
ActiveX-based component coordinator. OTMs provide an open

45

and "highly-toolable" application platform for the middle-tier. [Ofali]
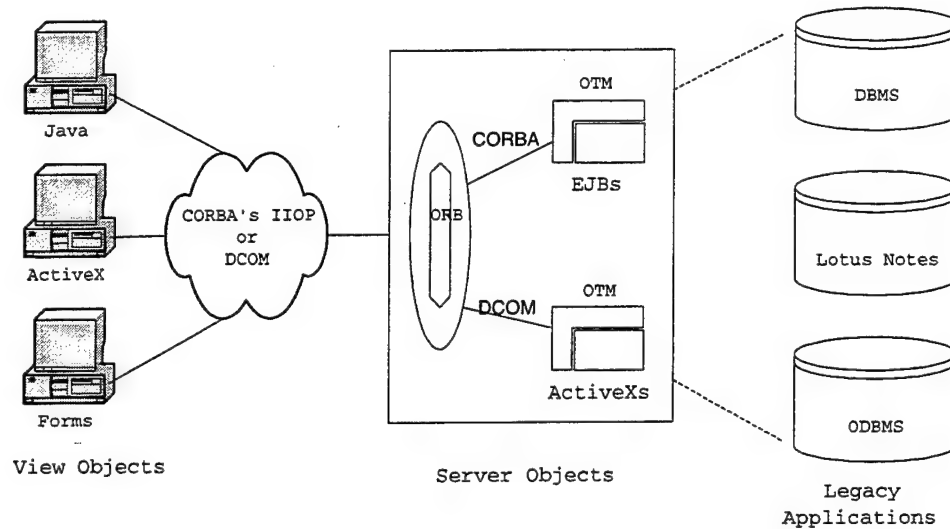
Figure 5: Integrating TP Monitors with ORBs

1.   **Microsoft Transaction Server (MTS)**

Microsoft Transaction Server (MTS) provides a server-centric environment for developing and deploying three-tiered applications based on Microsoft's COM technologies. As shown in Figure 6, the MTS architecture allows application logic components to run under the control of MTS on servers.

46
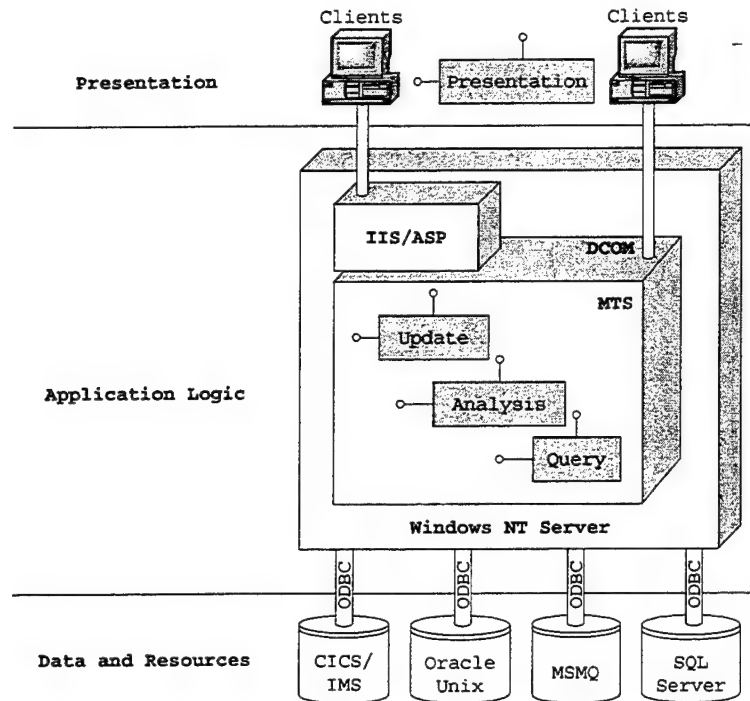
Figure 6: MTS Application Architecture

MTS applications are invoked by the presentation-
centric components running on clients via COM technologies.
Application logic components can access a number of
different databases, message queuing servers, CICS and IMS
applications. Access to databases and resources is done
through MTS Resource Dispensers that perform services such
as connection pooling automatically. MTS also supports
automatic transactions so that access to data and resources
is done with all-or-nothing protection. A simple MTS
application might consist of three processes running on the
same computer: an Excel spreadsheet, which is calling

47

methods in an MTS component, which in-turn accesses a SQL
Server database.

Within the MTS architecture, programmers build
presentation and application logic components with any tool
or programming language that can generate COM-compliant
DLLs. Microsoft specifies the following rules for
components to work with MTS:

- Components must create a reference to their MTS
  Context Object by making a simple API call. Creating
  the reference enables the component to take advantage
  of MTS services such as transaction and security
  support.

- Do not save state information across transaction
  boundaries within components (e.g., in local or global
  variables). Components that save state are less
  scalable, because MTS cannot recycle their resources
  when they finish executing. State should be kept in
  databases or in the Shared Property Manager (SPM) in
  MTS and retrieved by components when needed.

- When a component completes execution successfully, it
  must call the SetComplete method on the MTS Context
  Object. This tells MTS that this component wishes to
  commit any work it has performed when all components
  involved in the transaction finish executing. Calling
  SetComplete also tells MTS that it can recycle any
  resources held by the component.

- If a component cannot complete executing successfully,
  it must call the SetAbort method on the MTS Context
  Object. This tells MTS that it should abort the
  current transaction and roll back all changes made by
  components involved in the transaction. Calling
  SetAbort also tells MTS that it can recycle any
  resources held by the component. [Microsoft]

48

When components follow the rules listed, applications can take advantage of MTS benefits such as enhanced scalability, performance, and management with no additional development.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. NITES INTEGRATION DESIGN AND SPECIFICATIONS

A. INTRODUCTION

In developing an architectural framework for Commercial Of-The-Shelf (COTS)/Legacy Systems Integration, we developed the Software Requirements Specification (SRS) referenced in Appendix A. The Naval Integrated Tactical Environmental System I (NITES I) was used as a basis of the requirements for the architectural framework. The Client workstations were developed under Windows NT platform. Because of this constraint, we chosed to used DCOM. We found that DCOM was more suited for Microsoft-centric environment.

This section focuses on a high-level design of a distributed components architecture for COTS/Legacy Systems Integration.

B. NITES I SYSTEM OVERVIEW

The NITES I project is a Space and Naval Warfare (SPAWAR) sponsored project within the DoD. The driving force of the NITES I project is the integration of existing legacy systems with the latest industry-based COTS and Government-Off-the-Shelf (GOTS) technology. This includes the use of personal computers and COTS components.

The purpose of NITES I system is to provide
integration of METOC data with the necessary tools to
support the war fighter in the field. Basically, NITES I
acquire and assimilate various METOC data for access by the
US Navy and Marine Corps. The NITES I system is the
primary METOC data fusion platform and principal METOC
analysis workstation, intended to be operated on both a
classified and unclassified network environment by METOC
personnel. The system receives, processes, stores, and
disseminates METOC data and provides analysis tools to
render products for application to military operations.
Data and information are stored in a unified METOC database
on the Joint Command, Control, Communications, Computers,
Information, Surveillance and Reconnaissance (C4ISR)
network and are available to local and remote users.
Figure 7 displays an overview of the components of the
overall NITES architecture.
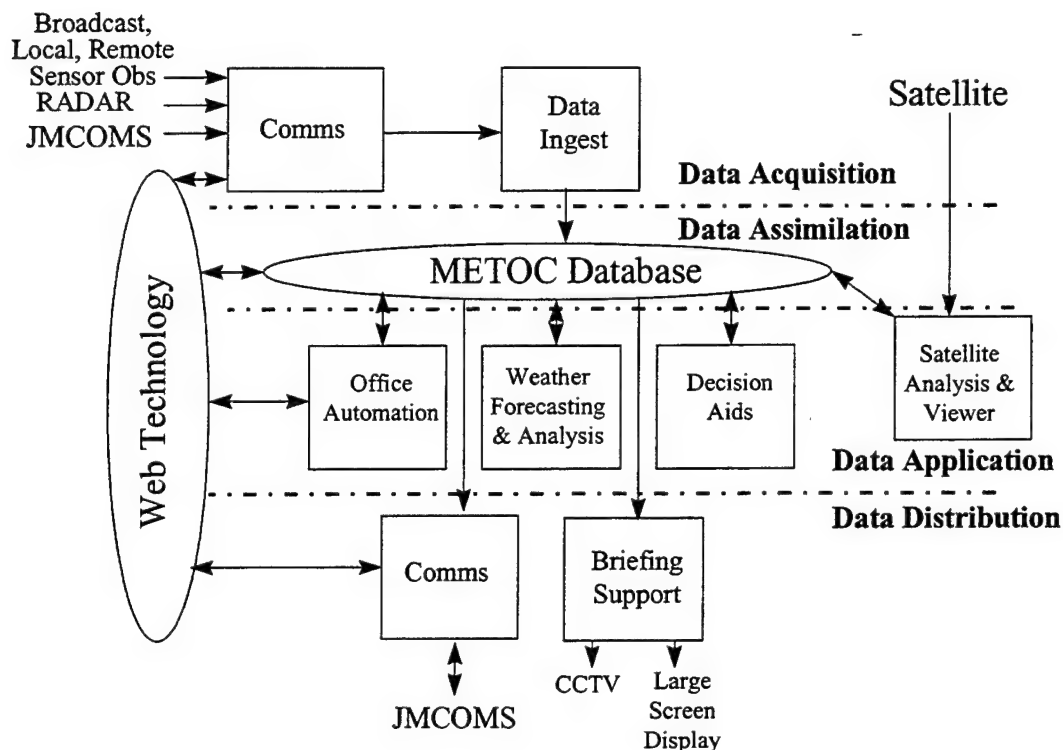
Figure 7: NITES I System Architecture Diagram

These components are grouped by functionality into:

1)   Data Acquisition:
      The NITES I system has to be able to receive
      various data from many different sources. Data
      types include point data, line data, surface or
      level data, volume data, imagery data, and text
      data.

2)   Data Assimilation:
      The NITES I system has to be able to process the
      data received and store them in the METOC
      database.

3)   Data Application:
      The METOC database can be accessed by
      applications software. Additionally, the User
      has visualization tools and a briefing package

53

that allow them to turn METOC information into
useful products for the Customer.

4)   Data/Product Dissemination:
       Numerical analysis and forecast model products
       and data, and forecaster generated METOC products
       and data are made available to Customers via Web
       technology, Closed-Circuit Television (CCTV),
       Local Area Network (LAN), or JMCOMS. [SPAWAR]
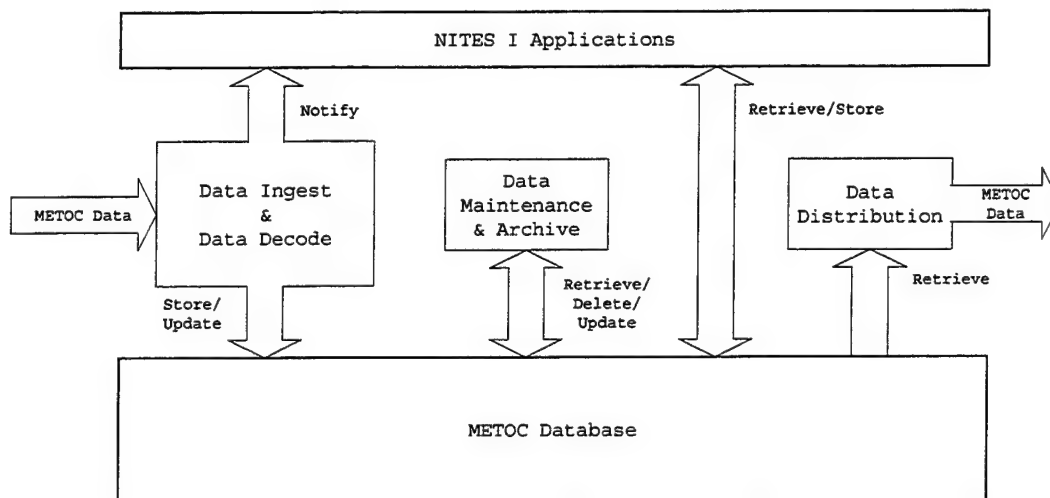
Figure 8: NITES I Data Flow

NITES I is intended to provide the user with METOC

data required to access and forecast the environment.   Its

backbone is the seamless transfer of METOC information from

raw data through regional centers to the warrior.   One of

the functions is to turn METOC data into useful products.

Table 1 shows the NITES data formats.

Table 1: NITES I Data Descriptions

Point Data

Data Description	Data Size	Data Retention	Data Source	Data Format	Comm Path	Data Frequency	Processing
Surface Observations Global	0.1-3KB	24 hr + 72 hr for selected stations (10 MIN)	Regional	WMO/BUFR	Fleet Multichannel Broadcast	100-300/hr	Station plot, contour, time series display and trends display for a given stations, Clipboard display
					High Speed Fleet Broadcast	300-800/hr	
			Regional, FNMOC,	WMO/BUFR	SIPRnet, NIPRnet, JMCOMS	800-2000/hr	
			International	WMO/BUFR	RATT	50/hr	
Surface Observations Local	0.2KB	1 week + archive	SMOOS, MORIAH, METMF	WMO/BUFR	LAN	Every 30-60 min	
			Manual	GUI	Keyboard Entry	1/hr	
TAF	0.3KB	24hr	Regional	WMO/BUFR	Fleet Multichannel Broadcast	50-100/hr	Plot, overlay, Clipboard display
					High Speed Fleet Broadcast	100-300/hr	
			International	WMO/BUFR	RATT	50/hr	
PIREP	0.1KB	24hr	Regional	WMO/BUFR	Fleet Multichannel Broadcast	50-100/hr	Plot Clipboard display
					High Speed Fleet Broadcast	100-300/hr	
			International	WMO/BUFR	RATT	50/hr	
Cloud Drift	0.2KB	24hr	Regional	WMO/BUFR	Fleet Multichannel Broadcast	50-100/hr	Plot, contour, calculate
					High Speed Fleet Broadcast	100-300/hr	
			International	WMO/BUFR	RATT	50/hr	
Navigation Data	4KB	Current Position	SMQ-11 (NAVSSI)	c-element	LAN	1/second	Used by SMOOS and SMQ-11

C. DISTRIBUTED COMPONENTS ARCHITECTURE DESIGN

The distributed components architectural design is based on a three-tiered architecture consisting of the presentation layer, logic layer, and database. In order to integrate COTS/Legacy systems with the existing architecture, we used wrappers. Wrappers allow next-generation systems to interact with wrapped legacy systems by modifying its interface. Once wrapped, legacy systems can participate in distributed object environment using object request brokers (ORBs). The following is a generic architecture containing the following: an Application object, Controller component, Application Wrapper component, Glue component, and the Database. The Application Wrapper transforms each COTS application into COM component interfaces, which may reside on external NT machines. Each application requires a wrapper object to handle the specific type of file that the application uses. The controller object monitors the application startup and references the appropriate wrapper component. The wrapper component will then translate the data accordingly so that the "glue" component may be able to store/retrieve data to/from the database.
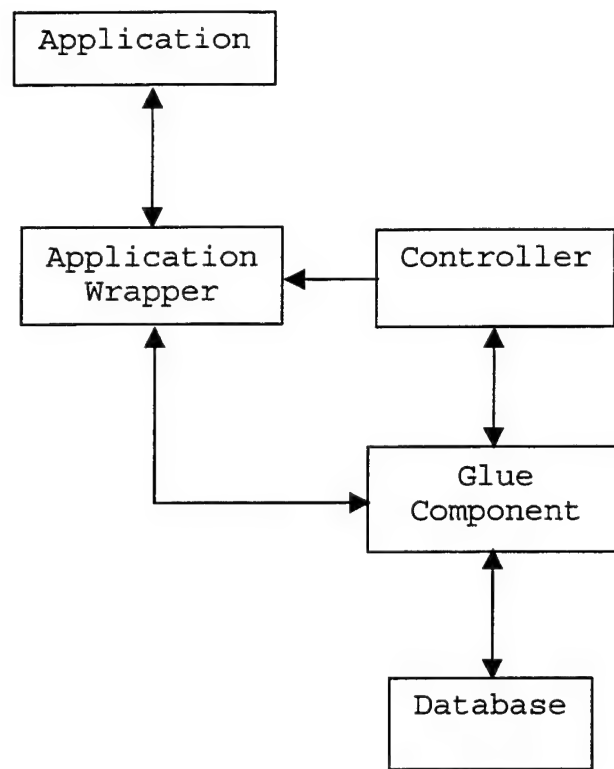
Figure 9: Component Integration Architectural Diagram

The following are generic functional descriptions of the various components:

Application

This is the COTS/Legacy Application to be wrapped to be able to interact with the rest of the components within the system.

System Controller

It manages the communications between the wrapper component and glue component. It signals events to all

components within the system.

Application Wrapper

This is the front-end proxy component, which provides
the methods interface of a particular application. It
reads data specific to application and formats data to be
stored to database. Wrappers are unique to specific
application based on the data types and how they are
referenced to the database. Remote legacy components are
encapsulated using their APIs (if available) to interact
with other components.

Glue Component

This is the data type management layer, which stores
and retrieves data to/from database using the provided
database APIs. It hides all the communications management
from the programmer.

In the next section we will adapt our generic design
to integrate an application with the NITES I system
architecture.

V.   NITES INTEGRATION CASE STUDY

Our goal in this section is to utilize the basis of the design presented in the previous section to integrate COTS/Legacy applications within the NITES subsystem architecture. The purpose is to interface with the application and distribute data to and from a relational database. The prototype developed for this case study focuses on distribution of imagery data (i.e. Jpg, Mif, Nif file formats). Our goal is to integrate the Continuous Brief Application to the NITES database architecture. The continuous brief application is in PowerPoint Slide Show format, presenting images on a continual basis. Figure 10 shows the diagram of the following objects: Continuous Brief Wrapper, Glue component, Local Imagery Directory, and System Controller.

CB App

CB App(n)

HTTP (IIS), DCOM (ActiveX control)

CBWrapper

CBWrapper n

IMAGE
EDITOR

Store New
Image Files

Notify image
updates

Requests
Data

System
Controller

polls

Local Image
Directory

Provides
Data

Notifies

Retrieve New
Image Files

Glue component

OBDC/ADO

retrieves

stores

Database
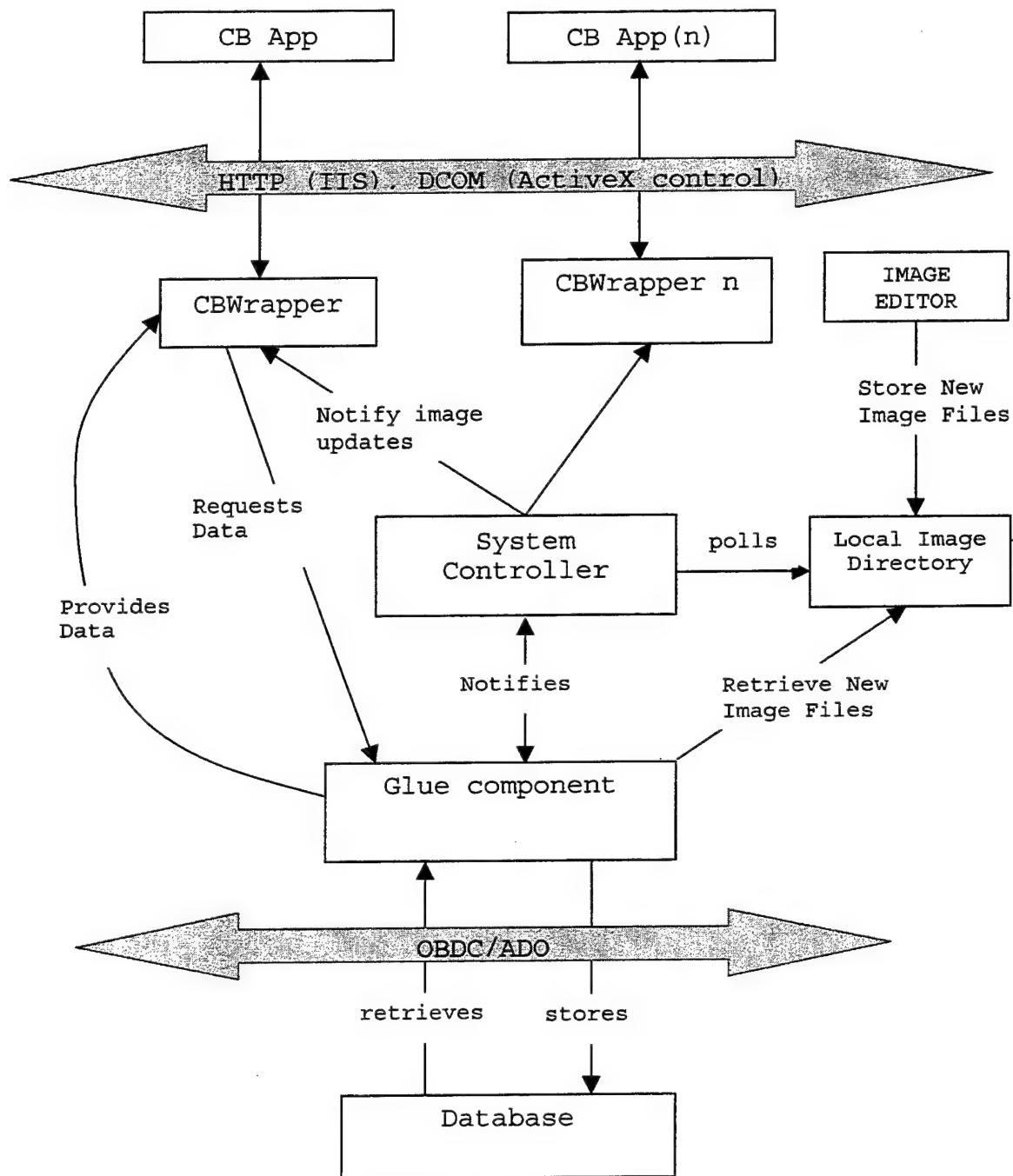
Figure 10: Continuous Brief Component Integration Diagram

Image Editor

Image Editor is an application which edits and saves an imagery file to the local image directory.

Local Image Directory

The local image directory was created to hold imagery files, which could be added by other external components.

Continuous Brief Wrapper (CBWrapper)

The Continuous Brief wrapper (CBWrapper) is an ActiveX component that accepts user inputs such as image type, images size, number of images, and display time interval. The CBWrapper registers with the system controller so that it will be notified of imagery updates. The Controller component uses the UpdateBrief event to notify the CBWrapper when to update the image brief. The CBWrapper uses the following PowerPoint APIs to display and update imagery data in a slide show fashion:

- Presentations.Add - Creates a presentation. Returns a Presentation object that represents the new presentation.
- Slides.Add - Creates a new slide and adds it to the collection of slides in the specified presentation. Returns a Slide object that represents the new slide.

61

- SlideShowTransition - Contains information about how the specified slide advances during a slide show.

- SlideShowSetting - Represents the slide show setup for a presentation.

- Shapes.AddPicture - Creates a picture from an existing file. Returns a Shape object that represents the new picture.

- Shapes.PictureFormat - Contains properties and methods that apply to pictures and OLE objects. The LinkFormat object contains properties and methods that apply to linked OLE objects only. The OLEFormat object contains properties and methods that apply to OLE objects whether or not they are linked.

System Controller

The System Controller consist of a Controller object and a Monitor object. The Controller is a COM object programmed to notify the CBWrapper and signals the wrapper(s) to update the brief(s). The Controller also passes information (image type) to the Glue component for storing data to the database. The Monitor object polls the local image directory for new imagery files. The Monitor object will raise an event to notify the controller object when new imagery files (by image type) arrive at the local

62

image directory. The controller object will then notify all wrapper objects registered for updates to update their briefs from the database. In the case where the components are on separate machines, the DCOMCNFG utility is used to set the location of each of these COM-based components and user account assigned to the components. The automation data types are used to make marshaling and un-marshaling of data transparent to each component, thus supporting a distributed objects environment.

Glue Component

The glue component, also referred to as the image wrapper, is a type of data wrapper when triggered, stores and retrieves imagery data from the database using appropriate database ActiveX Data Objects (ADO) calls. ADO 2.0 supports events, which are notifications that certain operations are about to occur or have occurred. There are two families of events: ConnectionEvent and RecordsetEvent. The Connection object issues ConnectionEvent events, and the Recordset object issues RecordsetEvent events. Events are processed by event handler routines, which are called before certain operations start or after such operations conclude. The following is a Visual Basic ADO connection example:

63

```
Dim WithEvent connEvent as Connection
Dim conn as New Connection
set connEvent = conn        'Turn on event support.
conn.Open(...)
...
set connEvent = Nothing     'Turn off event support.
...
Private Sub connEvent_ConnectComplete(ByVal err as ADODB.Error,
adStatus as ADODB.EventStatus, ByVal pConnectionas
ADODB.Connection)
'Check the error object only if adStatus equals
adStatusErrorsOccurred.
...
End Sub
```

ADO creates a connection object, but does not assign that object to an object variable. If multiple Recordset objects are created over the same connection, each connection object should be created and opened; this assigns the Connection object to an object variable. If the object variable is not used when opening the Recordset objects, ADO will create a new Connection object for each new Recordset, even the same connection string is passed. Once the data is committed to the database, the glue component disconnects and terminates the remote connection to the database. The Glue component will then raise the event to notify the Controller that it is done with the data storage.

Figure 11 shows the sequences of events when a new image file arrives in the image directory.

1. When a new imagery file arrives in the imagery directory.

64

2. The system controller notifies the glue
    component.

3. The glue Component stores the imagery data to the
    database.

4. The glue component notifies the system controller
    when the data is added to the database.

5. The controller notifies the CBWrapper of the
    update.

6. The CBWrapper sends a request to the glue
    component for the data.

7. The glue component retrieves the data from the
    database and passes it to the CBWrapper,

8. The CBWrapper automatically updates the
    Continuous Brief with new imagery data.

Image Directory	System Controller	CB Application	CBWrapper	Glue Component	Database

Poll app for new files

Notify controller new file added to database

New file arrives

Retrieve new file

Connect to database

Store image to database

Notify controller new file added to database

Disconnect from database

Notify wrapper of new file update

Pass request for new images to glue component

Connect to database

Retrieve image from database

Return requested images

Pass request for new images to CBWrapper

Disconnect from database

Updates Continuous Brief Display

Figure 11:   Imagery Updates Sequence Diagram

For non imagery data, the glue component uses XML-based format called Weather Observation Markup Format (OMF) for database storage and retrieval. METOC has developed APIs for automatic data storage and retrieval to support XML-based formats. [SPAWAR PMW-185]

The OMF contains the following elements:

- Reports – defines a group of weather observation reports

    o METAR for a single METAR report

    o SPECI for single SPECI report

    o UAR for a combined Rawinsonde and Pibal Observation report

    o BTSC for ocan profile data (temperature, salinity, current)

    o SYN for a surface synoptic report from a land or sea station

- Advisories – defines a collection of weather hazard warnings

    o SIGMET – SIGnificant METeorological Information

- Forecasts – defines a set of weather forecasts

    o TAF – Terminal Aerodrome Forecasts

- Messages – defines a set of plain-text bulletins.

The OMF data descriptions are shown in Appendix B tables 1-1 through 1-16.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS

A. RESULTS OF THE CASE STUDY

The integration of the Continuous Brief Application demonstrated the use of Microsoft's COM/DCOM to communicate between components, the use of wrappers to support COTS/Legacy components, and database access via API calls in COTS/Legacy systems integration. The user was able to select the number of images to be displayed in PowerPoint Slideshow format. In addition, new imagery files arriving in the image directory were updated in the Continuous Briefing display.

We found that DCOM was more suited for Microsoft-centric environments. Although a third party vendor, Software AG, has released a version of DCOM for Solaris UNIX, DCOM is best implemented as a solution where the environment is based on Microsoft products. Since the implementation was done under Windows NT this was not an issue.

B. SUMMARY OF THE THESIS RESEARCH

The Software Requirements Specifications and Software Design Specification were developed to support integration of COTS/Legacy software. Various distributed computing

69

technologies were researched to identify their capabilities and limitations. Because ORBs are relatively new, and considering the rapid evolution of software development, developers should determine whether the added cost of using ORB is acceptable for the given application, and whether the difficulty of learning and using ORBs is offset by the development time saved by not having to implement a subset of its features. This was a significant reason why we chose to use DCOM over CORBA, in addition to the realization that DCOM was better suited for the Microsoft platform environment.

C.   CONCLUSIONS

The Information Technology for the 21st Century (IT-21) directive will continue to be the driving force for the Navy and DoD to transition monolithic legacy systems to heterogeneous distributed systems. Distributed ORB technology such as COM/DCOM, and CORBA/JavaBeans are new solutions based on distributed objects which will provide software engineers and developers a method to manage communication and data exchange between objects. ORBs promote interoperability of distributed objects systems because they allow developers to build systems based on objects from different vendors independent of how it is

implemented. However, due to the unique requirements of the DoD, developers should have a thorough understanding of the ever-increasing industry-accepted solutions to distributed computing before implementing its architecture. As time goes by, and as the competition between different ORB technology increases, the shortcomings (i.e., performance and ease of use) of each methodology will be improved upon. This will allow developers to work towards reaching the goal of interoperability and global data distribution, so that war fighters in the theater can access mission-critical data anywhere around the world and with different platforms.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A: SOFTWARE REQUIREMENTS SPECIFICATION FOR AN ARCHITECTURAL FRAMEWORK OF DOD COTS/LEGACY SYSTEM

GENERAL DESCRIPTION

This appendix contains the Software Requirements Specification extracted from the SRS for An Architectural Framework of the COTS/Legacy System, which was developed as a framework for integrating COTS products with existing DoD legacy systems.

ARCHITECTURE GOALS

Integration

The architecture shall provide seamless integration of COTS components.

The architecture shall support middleware approach to bind data, information and COTS components.

Because evolution and upgrade of COTS components are outside the control of the system integrators, the architecture of the COTS/Legacy system shall have an adaptable component configuration to reduce the effort of testing and reintegration when upgrades or new COTS packages are introduced to the system.

Interoperability

COTS and legacy systems reside on multiple platforms. This architecture shall address both UNIX and PC-based platforms.

In order to achieve and maintain information superiority on the battlefield, the architectural framework for DoD COTS/Legacy systems shall have the capability to share, receive and transmit on heterogeneous networks and hardware devices.

The data displayed on each desktop must have a common view.

The exchange of data between two systems shall be with no loss of precision or other attributes, in an unambiguous manner, in a format understood by both systems, and in such a way that interpretation of the data is precisely the same.

The architecture shall support standard application program interfaces (APIs) to communicate.

Adopted Framework Technology

Java/C++, web technologies, open systems, application program interfaces, common operating environment, object and component technology, commercial products and standards are all important to the COTS/Legacy system architecture. The COTS/Legacy system architecture shall adopt the Object

Management Group (OMG) object model as well as OMG standards for object and distribution management.

The COTS/Legacy system shall adopt Interface Definition Language (IDL) as the language for expressing the syntax of the framework services.

The COTS/Legacy system architecture shall be expressed as UML class and package diagrams, with detailed component descriptions using IDL with English narrative to provide semantics.

Security

DoD tactical systems are normally classified to some security level. In building this architectural framework, the architecture shall address the DoD Trusted Computer System Evaluation Criteria (TCSEC) to at least the B2 security level.

The architecture shall include discretionary access control (DAC). Only single level classification systems shall be supported in this architecture (i.e. no multi-level security).

Network Security

The trend in DoD is for networked systems vice standalone monolithic systems and because most systems have some level of classification, this architecture shall address network security.

The architectural framework shall support a secure network.

The architectural framework shall support the network security mechanisms specific to the target architecture, including firewalls, routers, encryption, and proxy services.

Network Communications

The architectural framework shall support different network protocols (e.g. TCP/IP) and topologies dependent on the target architecture.

The application layer shall be able to execute a variety of data management commands without having knowledge of the data location, database, file type, operating system, network protocol, or platform location.

Development Language

The architectural framework shall support any development language that is supported by the legacy system as well as any development language that supports platform independence for newly developed code in the target architecture.

Assumptions and Dependencies

Assumption 1:   Legacy systems are monolithic and not modifiable.

Assumption 2: Legacy systems have some existing mechanism for interaction.

Assumption 3: There are varying degrees of COTS. To be considered COTS, the component cannot be modified.

Assumption 4: Reliability, performance, safety and security must be weighed in the target architecture.

TARGET ARCHITECTURE FUNCTIONS

Database

COTS software applications, which handle data, tend to have their own mechanism and structure for the storage of the data internal to the COTS application. The architectural framework shall support the central storage of data vice allowing each COTS application to store its own data.

The architecture shall support remote access to the database.

The COTS/Legacy architecture shall support a distribution mechanism (i.e. Common Object Request Broker Architecture (CORBA)) to store and share data in a distributed environment.

The architecture shall ensure updates in one set of data are consistently made throughout the rest of the database.

Security

Discretionary Access Control (DAC) COTS applications for the commercial market do not normally address security. Because this architecture applies to DoD software systems, in accordance with DoD security standards, the architecture shall support the display of the system's security classification (up to and including SECRET classification) on each of the display monitors as well as on all printouts.

The displayed classification shall be editable by the operator.

Graphical User Interface (GUI)

The target architecture shall include a GUI style guide. If a GUI style guide does not exist for the target architecture, UNIX platforms shall adhere to the MOTIF standard and PC platforms shall adhere to the X-windows standard.

ARCHITECTURE ATTRIBUTES

Performance Requirements

The architecture shall optimize the database access over a network. The architecture shall allow concurrent access of the database to multiple users.

Reliability Requirements

The target architecture shall use standard fault-
tolerance technologies.   COTS components shall be
replicated to maintain the reliability and availability
requirements of DoD systems.

While the data traverses throughout various
applications, to different platforms, through the network
and to/from database, it must remain consistent and not
suffer any degradation.

Design Constraints

Because many existing legacy systems reside on UNIX
platforms and the DoD has made a commitment to move towards
PC architecture, the architectural framework shall be
platform independent, supporting both UNIX and PC platforms
with the goal of moving towards a pure PC architecture.   It
is not required that all COTS/Legacy system components be
executable on both platforms but the data must be able to
be shared by components on different platforms.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B OMF DATA AND DOCUMENT TYPE DEFINITIONS

Table 1-1. Basic Attributes of an Observation in OMF

Attribute	Brief Description	Format	Description
TStamp	Time Stamp	unsigned integer	UTC time in seconds since the Epoch, 00:00:00 Jan 1, 1970 UTC. This is the value returned by a POSIX function time(2). Example: Tstamp='937507702'
TRange	Time Interval	a string of form "aaa, bbb", where aaa and bbb are unsigned integer numbers specifying the beginning and the end timestamps of the interval.	Timestamps are in seconds since the Epoch, 00:00:00 Jan 1, 1970 UTC. These are the values returned by a POSIX function time(2). Example: Trange='937832400, 937915200'
LatLon	Specification of a point on the globe	A string of a form "aaa.bbb, ccc.ddd", where aaa.bbb and ccc.ddd are signed floating point numbers	The latitude and. longitude, respectively, of a point on the globe, in whole and fractional degrees. The numbers are positive for Northern latitudes and Eastern longitudes, and negative for Southern latitudes and Western longitudes. The range of the numbers is [-90.0, 90.0] for latitudes, (-180.0, 180.0] for longitudes. Example: LatLon='32.433, -99.850'

Table 1-1. Basic Attributes of an Observation in OMF

(Cont.)

Attribute	Brief Description	Format	Description
LatLons	Specification of a sequence of points on the globe	a string of a form "lat1, lon1, lat2, lon2, latn, lonn" where each pair (lat1, lon1, etc.) are signed floating point numbers	A sequence of pairs of numbers, each pair giving the latitude and longitude of a single point in the sequence, in whole and fractional degrees. See the LatLon attribute above for more details. Example: LatLons='38.420, -111.125, 36.286, -111.492, 36.307, -112.630, 37.700, -113.223, 38.420, -111.125'
Bbox	Bounding box, which tells the latitudal and the longitudal spans of an area of the globe	A string of a form "lat-N, lon-W, lat-S, lon-E", where the lats and lons are signed floating-point numbers, in degrees	Specification of the bounding box for an area of interest. Here lat-N is the latitude of the Northern-most point of the area, lat-S is the latitude of the Southern-most point, lon-W is the longitude of the Western-most point of the area, and lon-E is the Eastern-most longitude. It is required that lat-N >= lat-S. The left-lon (lon-W) may however be greater than the right-lon lon-E). For example, a range of longitudes [-170,170] specifies the entire world but Indonesia. On the other end, the range [170, -170] includes Indonesia only. By the same token, [-10,10] pertains to a 21-degree longitude strip along the Greenwich meridian, while [10,-10] specifies the whole globe except for that strip. Example: Bbox='60.0, -120.0, 20.0, -100.0'
Bid	Station identification group	Unsigned integer	WMO Block Station ID, or other identifier for buoy or ship

82

Table 1-1. Basic Attributes of an Observation in OMF

(Cont.)

Attribute	Brief Description	Format	Description
SName	Call sign and full name of an observing station	A string of the form "ccccc, name", where ccccc are the call letters of the station (ICAO station id: 4 or 5 upper-case letters, may be omitted), name is an arbitrary string describing the station	The observing stations ICAO, aircraft, or ship call sign, plus a plain-text station name (e.g. "KMRY, Monterey CA Airport" Example: Sname='KYNL, YUMA (MCAS)'
Elev	Elevation	A non-negative integer, or omitted if unknown.	Station elevation relative to sea level, in meters. This attribute may specify a surface elevation of an observation station, or an upper-air elevation for an upper-air report. Example: Elev='16'

Table 1-2. OMF Attributes for METAR and SPECI Reports

Attribute	Brief Description	Format	Description	Req'd ?
TStamp	Time Stamp	<-------------See Table 1-1------------->		Yes
LatLon	Station latitude and longitude	<-------------See Table 1-1------------->		Yes
BId	Station Identification Group	Unsigned integer	WMO Block Station ID	Yes
SName	Call sign and full name of an observing station	<-------------See Table 1-1------------->		Yes
Elev	Station elevation	<-------------See Table 1-1------------->		No
Vis	Visibility	a number of meters, omitted, or a special token "INF"	Horizontal visibility in meters	No
Ceiling	Ceiling	a number of feet, omitted, or a special token "INF"	Ceiling in feet	No

84

Table 1-3. OMF Attributes for the SYN Element

Attribute	Brief Description	Format	Description	Req'd?
TStamp	Time Stamp	<-------------See Table 1-1------------->		Yes
LatLon	Station latitude and longitude	<-------------See Table 1-1------------->		Yes
BId	WMO Block Station Number	String	For a buoy or other observation platform, this id is a combination of a WMO region number, subarea number (per WMO Code Table 0161), and the buoy type and serial number. This information is reported in Section 0 of a synoptic report. If Section 0 contains a call sign rather than a numerical id (as typical with FM 13 SHIP reports), the BId attribute is computed as itoa(1000009 + hc) % 2^30, where hc is a numerical representation of the call letters considered as a number in radix 36 notation. For example, "0000" hashes to 0, and "ZZZZ" hashes to 1,679,615. Note this formula makes the BId attribute a unique numeric identifier for the station.	Yes
SName	Call sign and full name of an observing station	<-------------See Table 1-1------------->		Yes
Elev	Station elevation	<-------------See Table 1-1------------->		No

Table 1-3. OMF Attributes for the SYN Element (Cont.)

Attribute	Brief Description	Format	Description	Req'd ?
Title	Report title	String	Title defining type of report: AAXX (FM-12), BBXX (FM-13), or ZZYY (FM-18)	Yes
Stype	Station type	String	Type of station: automated (AUTO) or manned (MANN); defaults to MANN	No

Table 1-4. OMF Attributes for the SYG Element

Attribute	Brief Description	Format	Description	Req'd ?
T	Air Temperature	positive, zero, or negative number	Air temperature in degrees Celsius	No
TD	Dew point Temperature	positive, zero, or negative number	Dew point temperature in degrees Celsius	No
Hum	Relative humidity	non-negative number	Relative humidity in per cent	No
Tmm	Extreme temperatures over the last 24 hours	a string of a form "mmmm, MMMM" or omitted	Minimum and maximum temperatures (degrees Celsius) over the last 24 hours	No
P	Station pressure	positive number	Atmospheric pressure at station level, in hectoPascals	No
P0	Sea level pressure	positive number	Atmospheric pressure at station, reduced to sea level, in hPa	No
Pd	Pressure tendency	String of form "dddd", or omitted	Pressure tendency during the 3 hours preceding the observation	No
Vis	Visibility Number of meters, omitted, or a special token "INF"	Horizontal visibility in meters	Horizontal visibility in meters	No
Ceiling	Ceiling	Number of feet, omitted, or a special token "INF"	Ceiling in feet	No
Wind	Wind speed and direction	String of form "nnn, mm" or omitted	nnn is a true direction **from** which the wind is blowing, in degrees, or VAR if " the wind is variable, or all directions or unknown or waves confused, direction indeterminate." This is an integer number within [0,360), with 0 meaning the wind is blowing from true North, 270 stand for the wind blowing from due West. Normally this number has a precision of 10 degrees. mm is the wind speed in meters per second.	No

87

Table 1-4. OMF Attributes for the SYG Element (Cont.)

Attribute	Brief Description	Format	Description	Req'd?
Wx	Past and present weather conditions and phenomena	String of four digits, "NOSIG", or omitted	See WMO-306, Code tables 4677 and 4561 for the meaning of the four digits. This attribute is coded as "NOSIG" if there is no significant phenomenon to report. The attribute is omitted if not observed or data is not available (see ix indicator, Code table 1860).	No
Prec	Precipitation amount	String of form "nnn, hh" or "" or omitted	nnn is the amount of precipitation, which has fallen during the period preceding the time of observation. The precipitation amount is a non-negative decimal number, in mm. hh is the duration of the period in which the reported precipitation occurred, in whole hours. This attribute is encoded as "" if no precipitation was observed. The attribute is omitted if unknown or not available (see iR indicator, Code table 1819). Sea stations typically never report precipitation.	No
Clouds	Amounts and types of cloud cover	String of five symbols "tplmh" or omitted	The first digit is the total cloud cover in octas (Code table 2700). The second digit is the cloud cover of the lowest clouds, in octas. The other three symbols are types of low, middle, and high clouds, resp. See WMO-306 Code tables for more details.	No

Table 1-5. OMF Attributes for the SYSEA Element

Attribute	Brief Description	Format	Description	Req'd?
T	Sea surface temperature	Positive, zero, or negative number	Sea surface temperature in degrees Celsius	No
Wave	Sea wave period and height	String of form "pp, hh" or omitted	pp is the period of wind waves in seconds. hh is the height of wind waves, in meters. If a report carries both estimated and measured wind wave data, the instrumented information is preferred.	No
SDir	Ship's course and speed	String of form "nnn, mm" or omitted.	nnn is a true direction of resultant displacement of the ship during the three hours preceding the time of observation. The number is in degrees, or VAR if "variable, or all directions or unknown or waves confused, direction indeterminate." This is an integer number within [0,360), with 0 meaning the ship has moved towards the true North; 270 means the ship has moved to the West. Normally this number has a precision of 45 degrees. mm is the average speed made good during the three hours preceding the time of observation, in meters per second.	No

89

Table 1-6. OMF Attributes for the UALEVEL Element

Attribute	Brief Description	Format	Description	Req'd?
Ref	Reference to sounding Part	String - "TTAA", "TTBB", etc.	Reference to the part of the sounding from which the level data were derived	Yes
P	Pressure	positive number	Atmospheric pressure at sounding level, in hectoPascals	Yes
H	Geopotential height	Non-negative number of geopotential meters, or 'SURF' for surface, 'TROP' for tropopause, 'MAXW' for level of maximum winds, 'MAXWTOP' for maximum wind level at the top of the sounding, or omitted	Geopotential height of the reported level, or a special height indicator	No
T	Air Temperature	positive, zero, or negative number	Air temperature in degrees Celsius at the reported level	No
DP	Dew point temperature	positive, zero, or negative number	Dew point temperature in degrees Celsius at the reported level	No

Table 1-6. OMF Attributes for the UALEVEL Element (Cont.)

Attribute	Brief Description	Format	Description	Req'd?
Wind	Wind speed and direction	String of form "nnn, mm" or "nnn, mm bbb" or "nnn, mm ,aaa" or "nnn, mm bbb, aaa" or omitted	nnn is a true direction from which the wind is blowing, in degrees, or VAR if " the wind is variable, or all directions or unknown or waves confused, direction indeterminate." This is an integer number within [0,360), with 0 meaning the wind is blowing from true North, 270 stands for the wind blowing from due West. Normally this number has a precision of 10 degrees. mm is the wind speed in meters per second. If specified, bbb stands for the absolute value of the vector difference between the wind at a given level, and the wind 1 km below that level, in meters per second. The number aaa if given is the absolute value of the vector difference between the wind at a given level, and the wind 1 km above that level, in meters per second.	No

Table 1-7. OMF Attributes for the BTSC Element

Attribute	Brief Description	Format	Description	Req'd?
TStamp	Time Stamp	<---------- See Table 1-1 ---------->		Yes
LatLon	Latitude and Longitude of observation	<---------- See Table 1-1 ---------->		Yes
BId	Station identifier group	positive integer	For a buoy or other observation platform, this ID is a combination of a WMO region number, subarea number (per WMO-306 Code Table 0161), and the buoy type and serial number. This information is reported in Section 4 of a BTSC report. If Section 4 contains a call sign rather than a numerical id, the BId attribute is computed as itoa(1000009 + hc), where hc is a numerical representation of the call letters considered as a number in radix 36 notation. For example, "0000" hashes to 0, and "ZZZZ" hashes to 1,679,615. Note this formula makes the Bid attribute a unique numeric identifier for the station.	Yes
SName	Call sign	string	Ship's call sign, if reported	Yes
Title	Report type	string	"JJYY" - FM 63 X Ext. BATHY report "KKXX" - FM 64 IX TESAC report "NNXX" - FM 62 TRACKOB report	Yes
Depth	Water depth	positive number	Total water depth at point of observation	No

Table 1-8. OMF Attributes for the BTID Element

Attribute	Brief Description	Format	Description	Req'd?
DZ	Indicator for digitization	"7" or "8" or omitted	Indicator for method of digitization used in the report (k1 field). See WMO-306 Code Table 2262. Required for BATHY and TESAC reports	No
Rec	Instrument type code	5-digit code	Code for expendable bathythermograph (XBT) instrument type and fall rate (WMO-306 Code Table 1770)	No
WS	Wind speed units code	"0", "1", "2", "3", or omitted	Indicator for units of wind speed and type of instrumentation (iu field). See WMO-306, Code Table 1853.	No
Curr-s	Method of current speed measurement	"2", "3", "4", or omitted	Indicator for the method of current measurement (k5 field). See WMO-306 Code Table 2266.	No
Curr-d	Indicators for the method of subsurface current measurement	3-digit numerical code	Indicators for the method of subsurface current measurement (K6k4k3 codes). See WMO-306, Code Tables 2267, 2265, and 2264.	No
AV-T	Averaging period for sea temperature	"0", "1", "2", "3", or omitted (if no sea temperature data are reported)	Code for the averaging period for sea temperature (mT code). See WMO-306, Code Table 2604	No
AV-SAL	Averaging period for salinity.	"0", "1", "2", "3", or omitted (if no salinity data are reported)	Code for the averaging period for sea salinity (mS code). See WMO-306, Code Table 2604	No
AB-Curr	Averaging period for surface current direction and speed	"0", "1", "2", "3", or omitted (if no current data are reported)	Code for the averaging period for surface current direction and speed (mC code). See WMO-306, Code Table 2604	No
Sal	Method of salinity/depth measurement	"1", "2", "3", or omitted (if no salinity data are reported)	Code for the method of salinity/depth measurement (k2 code). See WMO-306, Code Table 2263.	No

Table 1-9. OMF Attributes for the BTAIR Element

Attribute	Brief Description	Format	Description	Req'd?
T	Air temperature	Positive, zero, or negative number, or omitted	Air temperature just above the sea surface, in degrees Celsius.	No
Wind	Wind vector	String of form "nnn,mm", or omitted	Here nnn is a true direction from which the wind is blowing, in degrees, or VAR if " the wind is variable, or all directions or unknown or waves confused, direction indeterminate." This is an integer number within [0,360), with 0 meaning the wind is blowing from the true North;, 270 means the wind is blowing from the West. Normally this number has a precision of 10 degrees. mm is the wind speed in meters per second.	No

Table 1-10. OMF Attributes for the BTLEVEL Element

Attribute	Brief Description	Format	Description	Req'd?
D	Depth	Non-negative number	Depth of the level in meters.	Yes
T	Water temperature	Positive, zero, or negative number, or omitted	Water temperature at the reported level.	No
S	Salinity	Positive number, or omitted	Salinity at the reported level, in parts per thousand.	No
C	Current vector String of form	"nnn,mm", or omitted	nnn is the true direction **toward** which the sea current is moving, in degrees, or VAR if "the current is variable, or all directions or unknown, direction indeterminate." This is an integer number within [0,360), with 0 meaning the current flows toward true North; 270 means the current is flowing toward the West. Normally this number has a precision of 10 degrees. mm is the speed of current in meters per second.	No

Table 1-11. OMF Attributes for the TAF Element

Attribute	Brief Description	Format	Description	Req'd?
TStamp	Time Stamp	<---------- See Table 1-1 --------->		Yes
LatLon	Latitude and Longitude of observation	<---------- See Table 1-1 --------->		Yes
BId	Block Station ID	positive integer	WMO Block Station ID of the reporting station	Yes
SName	Call sign	string	Ship's call sign, if reported	Yes

Table 1-12. OMF Attributes for the SIGMET Element

Attribute	Brief Description	Format	Description	Req'd?
class	SIGMET type	"CONVECTIVE", "HOTEL", "INDIA", "UNIFORM", "VICTOR", "WHISKEY"	Identifier for the type of SIGMET message	Yes
id	Identifier for a particular advisory	String	Identifier for the advisory; value depends on the advisory class.	Yes
TStamp	Time Stamp	<---------- See Table 1-1 ---------->		Yes
BBox	Bounding box for advisory area	<---------- See Table 1-1 ---------->		Yes

Table 1-13. OMF Attributes for the EXTENT Element

Attribute	Brief Description	Format	Description	Req'd?
Shape	Type of area specification	"AREA", "LINE", "POINT"	Type of area shape specified	Yes
LatLons	List of latitudes and longitudes defining the area	Positive, zero, or negative numbers in lat/lon pairs	Control points (vertices) for a polygon/polyline representing the affected area	Yes

Table 1-14. OMF Attributes for the MSG Element

Attribute	Brief Description	Format	Description	Req'd?
id	Message identifier	A NMTOKEN, a four-to-six-character string of a form T1T2A1A2ii	Designator for the message type and subtype (T1T2), area (A1A2), and sequence code (ii) of the message, as described in WMO-386.	Yes
Type	Message type	2-letter string (T1T2)	Designator for the message type and subtype (T1T2) as specified in WMO-386, Tables A and B1 through B6	Yes
TStamp	Time Stamp	<---------- See Table 1-1 ---------->		Yes
SName	Originating station name	String	String containing the identification of the station that originated the message (normally its ICAO call sign)	Yes
BBB	Annotation group	3-character string	So-called "BBB groups" from the abbreviated message line. They indicate that the message has been delayed, corrected or amended. A BBB group can also be used for segmentation. See the WMO-386 for more detail.	No
Descr	Description	String	Keywords and other information describing the message.	No
BBox	Bounding box	<---------- See Table 1-1 ---------->		No

Table 1-15 Layer Parameter Codes

Layer	Description	Example
adiabatic-cond	Adiabatic condensation level (parcel lifted from surface)	(layer adiabatic-cond)
atm-top	Level of the top of the atmosphere	(layer atm-top)
cloud-base	Cloud base level	(layer cloud-base)
cloud-top	Cloud top level	(layer cloud-top)
conv-cld-base	Level of bases of convective clouds	(layer conv-cld-base)
conv-cld-top	Level of tops of convective clouds	(layer conv-cld-top)
entire-atm	Entire atmosphere	(layer entire-atm)
entire-ocean	Entire ocean	(layer entire-ocean)
height	Height above ground (meters)	(layer height 1500)
height-between	Layer between two heights above ground in hundreds meters (followed by top and bottom level values)	(layer height-between 50 30) for layer between 5000 and 3000 meters above ground
height-between-ft	Layer between two heights above ground, in feet (followed by top and bottom level values)	(layer height-between-ft 15000 10000)
height-ft	Height above ground (feet)	(layer height-ft 50)
high-cld-base	Level of high cloud bases	(layer high-cld-base)
high-cld-top	Level of high cloud tops	(layer high-cld-top)
hybrid	Hybrid level (followed by level number)	(layer hybrid 1)
hybrid-between	Layer between two hybrid levels (followed by top and bottom level numbers)	(layer hybrid 2 1)
isobar	Level of an isobaric surface (followed by the isobar value of the surface in hectoPascals (hPa) (1000, 975, 950, 925,900,850,800,750,700, 650,600,550,500,450,400, 350,300,250,200, 150,100, 70, 50,30, 20,10)	(layer isobar 500)
isobar-between	Layer between two isobaric surfaces (followed by top and bottom isobar values in kPa, separated by a space)	(layer isobar-between 50 100) for layer between 500 and 1000 hPa

Table 1-15 Layer Parameter Codes (Cont.)

Layer	Description	Example
isobar-between-mp	Layer between two isobaric surfaces, mixed precision (followed by pressure of top in kPa and 1100 minus pressure of bottom in hPa)	(layer isobar-between-mp 50 100) for layer between 500 and 1000 hPa
isobar-between-xp	Layer between two isobaric surfaces, extra precision (followed by top and bottom isobar values expressed as 1100 hPa-isobar level, separated by a space)	(layer isobar-between 600 100) for layer between 500 and 1000 hPa
isotherm-0	Level of the zero-degree (Celsius) isotherm (or freezing level)	(layer isotherm-0)
land-depth	Depth below land surface in centimeters	(layer land-depth 5.0)
land-depth-between	Layer between two depths in ground (followed by the depth of the top of the layer and the depth of the bottom of the layer centimeters)	(layer land-depth-between 0 30) for layer from ground surface to 30 cm depth
land-height-cm	Height level above ground (high precision) (followed by height in centimeters)	(layer land-height-cm 50)
land-isobar	Pressure above ground level in hPa	(layer land-isobar 500)
land-isobar-between	Layer between two isobars abive levels (followed by top and bottom isobaric levels in hPa)	(layer land-isobar-between 500 1000)
low-cld-base	Level of low cloud bases	(layer low-cld-base)
low-cld-top	Level of low cloud tops	(layer low-cld-top)
max-wind	Level of maximum wind	(layer max-wind)
mid-cld-base	Level of middle cloud bases	(layer mid-cld-base)
mid-cld-top	Level of middle cloud tops	(layer mid-cld-top)
msl	Mean sea level	(layer msl)
msl-height	Height above mean sea level (in meters)	(layer msl-height 50)
msl-height-between	Layer between two heights above mean sea level in hundreds of meters (followed by top and bottom height values)	(layer msl-height-between 10 5) for layer between 1000 and 500 meters above ground
msl-height-ft	Height above mean sea level (in feet)	(layer msl-height-ft 5000)
sea-bottom	Bottom of the ocean	(layer sea-bottom)

99

Table 1-15 Layer Parameter Codes (Cont.)

Layer	Description	Example
sea-depth	Depth below the sea surface (meters)	(layer sea-depth 50)
sigma	Sigma level in 1/10000	(layer sigma 9950) for sigma level .995
sigma-between	Layer between two sigma surfaces (followed by top and bottom sigma values expressed in 1/100, separated by a space)	(layer sigma-between 99.5 100.0) for layer between .995 and 1.0
sigma-between-xp	Layer between two sigma levels (followed by top and bottom sigma values expressed as 1.1-sigma)	(layer sigma-between-xp .105 .100) for layer between .995 and 1.0
surface	Earth's surface	(layer surface)
theta	Isentropic (theta) level (followed by potential temperature in degrees K)	(layer theta 300)
theta-between	Layer between two isentropic surfaces (followed by top and bottom values expressed as 475-theta in degrees K)	(layer theta-between 150 200)
tropopause	Level of tropopause (top of troposphere)	(layer tropopause)

Table 1-16 PowerPoint API Function Description Table

Method	Description	Example
Application	Represents the entire Microsoft PowerPoint application.	MyPath = **Application**.Path
ActivePresentation	Returns a Presentation object that represents the presentation open in the active window. (Read-only)	Application **.ActivePresentation**.SaveAs MyPath
Presentations	Returns a Presentation object that represents the presentation in which the specified document window or slide show window was created. (Read-only)	firstPresSlides = Windows(1).Presentation.Slides.Count Windows(2).**Presentation**.PageSetup _ .FirstSlideNumber = firstPresSlides + 1
Presentations.Add	Creates a presentation. Returns a Presentation object that represents the new presentation.	This example creates a presentation, adds a slide to it, and then saves the presentation. With Presentations.**Add** .Slides.Add 1, ppLayoutTitle .SaveAs "Sample" End With
Slides	A collection of all the Slide objects in the specified presentation.	Use the Slides property to return a Slides collection: ActivePresentation.Slides.Add 2, ppLayoutBlank
Slides.Add	Creates a new slide and adds it to the collection of slides in the specified presentation. Returns a <u>Slide</u> object that represents the new slide.	This example adds a blank slide at the end of the active presentation. With ActivePresentation.Slides .**Add** .Count + 1, ppLayoutBlank End With
Shapes	A collection of all the Shape objects on the specified slide. Each Shape object represents an object in the drawing layer, such as an AutoShape, freeform, OLE object, or picture.	Use the Shapes property to return the Shapes collection. The following example selects all the shapes on myDocument. Set myDocument = ActivePresentation.Slides(1) myDocument.Shapes.SelectAll
Shapes.AddPicture	Creates a picture from an existing file. Returns a <u>Shape</u> object that represents the new picture.	Set myDocument = ActivePresentation.Slides(1) myDocument.Shapes.**AddPicture** "c:\microsoft office\" & _ "clipart\music.bmp", True, True, 100, 100, 70, 70

PowerPoint API Function Description Table (Cont.)

Method	Description	Example

Shapes.PictureFormat	Contains properties and methods that apply to pictures and OLE objects. The LinkFormat object contains properties and methods that apply to linked OLE objects only. The OLEFormat object contains properties and methods that apply to OLE objects whether or not they're linked.	Set myDocument = ActivePresentation.Slides(1)
		With myDocument.Shapes(1).PictureFormat
		    .Brightness = 0.3
		    .Contrast = 0.7
		    .ColorType = msoPictureGrayScale
		    .CropBottom = 18
		End With

SlideShowTransition	Contains information about how the specified slide advances during a slide show.	With ActivePresentation.Slides(1).SlideShowTransition .Speed = ppTransitionSpeedFast End With
SlideShowSetting	Represents the slide show setup for a presentation.	With ActivePresentation.SlideShowSettings .RangeType = ppShowSlideRange End With

The Complete OMF DTD

```
<!-- <!DOCTYPE OMF SYSTEM "OMF.dtd" [ -->

<!-- Weather Observation Definition Format DTD -->

<!-- This is the OMF XML DTD. It can be referred to using the

formal public identifier

-//METNET//OMF 1.0//EN

For description, see OMF.html

$Id: OMF.dtd,v 3.8 1999/10/25 18:18:31 oleg Exp oleg $

-->

<!-- Weather Observation Definition Format -->

<!-- Basic attributes -->

<!ENTITY % TStamp-type "NMTOKEN">

<!ENTITY % TRange-type "CDATA">

<!ENTITY % TStamp "TStamp %TStamp-type; #REQUIRED">

<!ENTITY % TRange "TRange %TRange-type; #REQUIRED">

<!ENTITY % LatLon "LatLon CDATA #REQUIRED">

<!ENTITY % LatLons "LatLons CDATA #REQUIRED">

<!ENTITY % BBox-REQD "BBox CDATA #REQUIRED">

<!ENTITY % BBox-OPT "BBox CDATA #IMPLIED">

<!ENTITY % BId "BId NMTOKEN #REQUIRED">

<!ENTITY % SName "SName CDATA #REQUIRED">

<!ENTITY % Elev "Elev NMTOKEN #IMPLIED">

<!-- Basic elements -->

<!ELEMENT VALID (#PCDATA)>

<!ATTLIST VALID %TRange;>

<!-- A collection of weather observation reports -->

<!ELEMENT Reports ( METAR | SPECI | UAR | BTSC | SYN )*>
```

```
<!ATTLIST Reports %TStamp;>

<!-- Common report attributes -->

<!ENTITY % ReportAttrs

"%TStamp; %LatLon; %BId; %SName; %Elev;

Vis NMTOKEN #IMPLIED

Ceiling NMTOKEN #IMPLIED

">

<!-- METAR and SPECI reports -->

<!ELEMENT METAR (#PCDATA)>

<!ATTLIST METAR %ReportAttrs;>

<!ELEMENT SPECI (#PCDATA)>

<!ATTLIST SPECI %ReportAttrs;>

<!-- A collection of weather hazard advisories -->

<!ELEMENT Advisories ( SIGMET | AIRMET | WW )* >

<!ATTLIST Advisories %TStamp;>

<!-- A SIGMET advisory -->

<!ELEMENT SIGMET (VALID, AFFECTING?, EXTENT, BODY) >

<!ATTLIST SIGMET

class (CONVECTIVE| HOTEL| INDIA| UNIFORM| VICTOR| WHISKEY) #REQUIRED

id NMTOKEN #REQUIRED

%TStamp;

%BBox-OPT;

>

<!ELEMENT AFFECTING (#PCDATA)>

<!ELEMENT EXTENT (#PCDATA)>

<!ATTLIST EXTENT

Shape (AREA| LINE| POINT) #REQUIRED

%LatLons;

>
```

```
<!ELEMENT BODY (#PCDATA)>

<!-- A collection of weather forecasts -->

<!ELEMENT Forecasts ( TAF )* >

<!ATTLIST Forecasts %TStamp;>

<!-- A Terminal Aerodrome Forecast -->

<!ELEMENT TAF ( VALID, PERIOD+ ) >

<!ATTLIST TAF

%TStamp; %LatLon; %BId; %SName;

>

<!ELEMENT PERIOD ( PREVAILING, VAR* )>

<!ATTLIST PERIOD

%TRange;

Title NMTOKEN #IMPLIED

>

<!ELEMENT PREVAILING (#PCDATA)>

<!ELEMENT VAR (#PCDATA)>

<!ATTLIST VAR

%TRange;

Title CDATA #REQUIRED

>

<!-- Rawinsonde and Pibal Observation reports -->

<!ELEMENT UAR ( UAPART+, UAID*, UACODE*, UALEVELS ) >

<!ATTLIST UAR

%TStamp; %LatLon; %BId; %SName; %Elev;

>

<!ELEMENT UAPART (#PCDATA)>

<!ATTLIST UAPART

id NMTOKEN #REQUIRED

>
```

```
<!ENTITY % UARef "Ref NMTOKEN #REQUIRED">

<!ELEMENT UAID (#PCDATA)>

<!ATTLIST UAID %UARef; >

<!ELEMENT UACODE (#PCDATA)>

<!ATTLIST UACODE %UARef; >

<!ELEMENT UALEVELS (UALEVEL)*>

<!ELEMENT UALEVEL (#PCDATA)>

<!ATTLIST UALEVEL

%UARef;

P NMTOKEN #REQUIRED

H NMTOKEN #IMPLIED

T NMTOKEN #IMPLIED

DP NMTOKEN #IMPLIED

Wind CDATA #IMPLIED

>

<!-- Bathythermal, Salinity and Ocean Currents Observations -->

<!ELEMENT BTSC ( BTID, BTCODE?, BTLEVELS ) >

<!ATTLIST BTSC

%TStamp; %LatLon; %BId; %SName;

Title (JJYY | KKXX | NNXX) #REQUIRED

Depth NMTOKEN #IMPLIED

>

<!ELEMENT BTID (#PCDATA)>

<!ATTLIST BTID

DZ (7|8) #IMPLIED

Rec NMTOKEN #IMPLIED

WS (0|1|2|3) #IMPLIED

Curr-s (2|3|4) #IMPLIED

Curr-d NMTOKEN #IMPLIED
```

```
AV-T (0|1|2|3) #IMPLIED

AV-Sal (0|1|2|3) #IMPLIED

AV-Curr (0|1|2|3) #IMPLIED

Sal (1|2|3) #IMPLIED

>

<!ELEMENT BTCODE (#PCDATA)>

<!ELEMENT BTLEVELS (BTAIR?, (BTLEVEL)*)>

<!ELEMENT BTAIR (#PCDATA)>

<!ATTLIST BTAIR

T NMTOKEN #IMPLIED

Wind CDATA #IMPLIED

>

<!ELEMENT BTLEVEL (#PCDATA)>

<!ATTLIST BTLEVEL

D NMTOKEN #REQUIRED

T NMTOKEN #IMPLIED

S NMTOKEN #IMPLIED

Curr CDATA #IMPLIED

>

<!-- Surface Synoptic Reports from land and sea stations -->

<!ELEMENT SYN ( SYID, SYCODE?, SYG?, SYSEA? ) >

<!ATTLIST SYN

%TStamp; %LatLon; %BId; %SName; %Elev;

Title (AAXX | BBXX | ZZYY) #REQUIRED

SType (AUTO | MANN) "MANN"

>

<!ELEMENT SYID (#PCDATA)>

<!ATTLIST SYID

WS (0|1|3|4) #IMPLIED
```

107

```
>

<!ELEMENT SYCODE (#PCDATA)>

<!ELEMENT SYG (#PCDATA)>

<!ATTLIST SYG

T NMTOKEN #IMPLIED

TD NMTOKEN #IMPLIED

Hum NMTOKEN #IMPLIED

Tmm CDATA #IMPLIED

P NMTOKEN #IMPLIED

P0 NMTOKEN #IMPLIED

Pd NMTOKENS #IMPLIED

Vis NMTOKEN #IMPLIED

Ceiling NMTOKEN #IMPLIED

Wind CDATA #IMPLIED

WX CDATA #IMPLIED

Prec CDATA #IMPLIED

Clouds CDATA #IMPLIED

>

<!ELEMENT SYSEA (#PCDATA)>

<!ATTLIST SYSEA

T NMTOKEN #IMPLIED

Wave CDATA #IMPLIED

SDir CDATA #IMPLIED

>

<!-- Plain-text WMO Meteorological messages -->

<!ELEMENT Messages ( MSG )* >

<!ATTLIST Messages %TStamp;>

<!ELEMENT MSG ANY >

<!ATTLIST MSG
```

```
id NMTOKEN #REQUIRED

Type NMTOKEN #IMPLIED

%TStamp;

%SName;

%BBox-OPT;

BBB CDATA #IMPLIED

Descr CDATA #IMPLIED

>

<!-- ]> -->
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C: CBWRAPPER/CONTROLLER/GLUE SOURCE CODE

```
'##########################################################
'#   File: WebInterface.ctl
'#   Date                 Author            History
'#   5/31/2000            Tam Tran          Created.
'##########################################################
Option Explicit
'*********************************************************
'
' The Continuous Brief wrapper (CBWrapper) is an ActiveX
' Control that represents the Graphical User Interface
' (GUI) via the Web browser (Internet Explorer). It allows
' an user to select the type of images that he/she wants
' to view. Also, it allows the user to set the number of
' images, the size, and the duration for the display.
'
'*********************************************************
Private mControllerConnector As ControllerConnector
Private mMonitor As Monitor
Private mMonitorConnector As MonitorConnector
Private WithEvents mController As Controller
' Get reference to Application object from the PowerPoint API.
Public myPPT As PowerPoint.Application
Public AppRunning As Boolean
Private BriefStarted As Boolean
Private downloadFolder As String
Private cfgFolder As String
Private ServerURL As String


'*********************************************************
'
' Reset the Continuous Brief GUI to its default values.
' Set slide show to fullscreen size.
' Set number of images to 24
' Set duration of the slide show to 0.
'
'*********************************************************
Private Sub Default_Click()
    ImageType.Text = "Select an image type"
    ImagesText.Text = "24"
    HeightText.Text = "540"
    WidthText.Text = "720"
    DurationText.Text = "0"
End Sub

'*********************************************************
'
' Update the brief.
' Use the GetImageDir method from the Controller object
' to get the location of the files.
' Use the Controller_UpdateBrief method to update the brief.
'
'*********************************************************
Private Sub Start_Click()
    Dim imageloc As String
```

111

```vb
        BriefStarted = True
        Call mController_UpdateBrief(ImageType.Text)
End Sub

'********************************************************
'
' Stop the slide show.
' Terminate the background running PowerPoint application.
' Free up the un-used object.
' Reset the AppRunning flag to false.
'
'********************************************************
Private Sub Stop_Click()
    If AppRunning Then
        myPPT.ActivePresentation.Close
        myPPT.Quit
        Set myPPT = Nothing
        AppRunning = False
        BriefStarted = False
    End If
End Sub

'********************************************************
'
' Initialize references to the Monitor and Controller objects.
'
'********************************************************
Private Sub UserControl_Initialize()

    Set mControllerConnector = New ControllerConnector
    Set mController = mControllerConnector.Controller
    Set mMonitorConnector = New MonitorConnector
    Set mMonitor = mMonitorConnector.Monitor
    AppRunning = False
    BriefStarted = False

    ' Add image types to the drop-box in the Continuous Brief GUI
    Dim intFile As Integer    ' FreeFile variable
    Dim inputStr As String
    Dim cfgFile As String
    Dim typeStr As String
    Dim locationStr As String
    Dim virtualDirStr As String
    Dim tmpFolderStr As String
    Dim tmpFileStr As String
    Dim downloadFileStr As String

    ' Set values for the URL, download folder, and a temporary filename
    ' %%%%%%%%%%%%%%%%%%%%%%
    ' Change config here:
    ServerURL = "http://isdsclient/" '"http://tampc.spawar.navy.mil/"
    ' %%%%%%%%%%%%%%%%%%%%%%%
    cfgFile = "cbdata.cfg"
    downloadFolder = Environ("TEMP") & "\cbdownload"
    cfgFolder = downloadFolder & "\cbdata"
    tmpFileStr = cfgFolder & "\" & cfgFile
    downloadFileStr = ServerURL & "/" & cfgFile
```

```
    ' Create a temporary directory for downloading data _
    Call createFolder(downloadFolder)
    Call createFolder(cfgFolder)
    Call downloadFile(downloadFileStr, tmpFileStr)

    intFile = FreeFile()
    Open tmpFileStr For Input As #intFile
    Do While Not EOF(intFile)
        Line Input #intFile, inputStr
        Call lineInfo(inputStr, typeStr, locationStr, virtualDirStr)
        ImageType.AddItem typeStr
    Loop
    Close #intFile
End Sub


'*********************************************************
' Receive Controller event to do the update for the brief.
' Parameters:
'       in: DataType - the data (images) type
'       in: imageDir - the directory where to find the images.
'
'*********************************************************
Private Sub mController_UpdateBrief(DataType As String)

    ' Check for the right type of data that the CBWrapper is showing.
    If (StrComp(ImageType.Text, DataType, vbTextCompare) = 0) And
BriefStarted Then
        Dim virtualDir As String
        Dim fileListName As String
        Dim tmpFileStr As String
        Dim tmpURLStr As String
        Call mController.GetImageInfo(ImageType.Text, ImagesText.Text,
_
                                          virtualDir, fileListName)
        ' Local variables declarations
        Dim myArray() As String
        Dim myPres As Presentation
        Dim fs, f, fc, f1, i, j, k
        Dim s As Slide
        Dim LeftVal As Long
        Dim TopVal As Long
        Dim imageW As Long
        Dim imageH As Long
        Dim ImgFile As String
        Dim intFile As Integer
        Dim inputStr As String

        ' Download the list of image filenames from server
        tmpURLStr = ServerURL & virtualDir & "/CB_listfile/" &
fileListName
        tmpFileStr = cfgFolder & "\" & fileListName
        Call downloadFile(tmpURLStr, tmpFileStr)

        ' Download image files from server
        intFile = FreeFile()
```

113

```
            Open tmpFileStr For Input As #intFile
            Do While Not EOF(intFile)
                Line Input #intFile, inputStr
                tmpURLStr = ServerURL & virtualDir & "/" & inputStr
                tmpFileStr = downloadFolder & "\" & inputStr
                Call downloadFile(tmpURLStr, tmpFileStr)
            Loop
            Close #intFile

            ' Get reference to the PowerPoint Application object.
            On Error Resume Next
            Set myPPT = GetObject(, "PowerPoint.application")
            If Err.Number <> 0 Then
                Set myPPT = CreateObject("PowerPoint.application")
            End If

            ' Set the AppRunning flag so that it will be
            ' checked when the STOP button is clicked.
            AppRunning = True

            ' Stop the current running slide show (if any)
            If myPPT.Presentations.Count <> 0 Then
                myPPT.ActivePresentation.Close
            End If

            ' Create new presentation with the new update data
            Set myPres = myPPT.Presentations.Add(True)

            ' Create a FileSystemObject for manipulating the file system
            Set fs = CreateObject("Scripting.FileSystemObject")
            Set f = fs.GetFolder(downloadFolder)
            Set fc = f.Files
            i = 1
            k = 1

            ' Store all filenames from the image directory
            ' to an array for sorting purpose.
            ReDim myArray(1 To fc.Count)
            For Each f1 In fc
                myArray(i) = f1.Name
                i = i + 1
            Next
            ' Sort the array.
            Call mMonitor.dhBubbleSort(myArray)

            ' Calculate the positions and dimensions for the images.
            Call GetDimensions(LeftVal, TopVal, imageW, imageH)

            ' Add the images to the PowerPoint presentation.
            For j = (fc.Count - ImagesText.Text + 1) To fc.Count
                    ImgFile = downloadFolder & "\" & myArray(j)
                    myPres.Slides.Add k, ppLayoutBlank
                    myPres.Slides.Item(k).Shapes.AddPicture ImgFile, True,
True, _
                                                        LeftVal,
TopVal, imageW, imageH
                    k = k + 1
```

114

```
            Next
            'Free up the FileSystemObject when done
            Set fs = Nothing
            Set f = Nothing
            Set fc = Nothing

            ' Configure the slide show properties and run the show
            For Each s In myPPT.ActivePresentation.Slides
                With s.SlideShowTransition
                    .AdvanceOnTime = True
                    .AdvanceTime = DurationText.Text
                End With
            Next

            With myPPT.ActivePresentation.SlideShowSettings
                .StartingSlide = 1
                .EndingSlide = ImagesText.Text
                .AdvanceMode = ppSlideShowUseSlideTimings
                .LoopUntilStopped = True
                .Run
            End With

        ' Delete the images when done creating the brief
        For i = 1 To fc.Count
            If fs.FileExists(downloadFolder & "\" & myArray(i)) Then
                Set f = fs.DeleteFile(downloadFolder & "\" & myArray(i),
True)
            End If
        Next
        End If
End Sub




'**********************************************************
' The GetDimensions subroutine calculates the positions
' (Left, Top), and the dimensions (Height, Width)
' for the images.
' Parameters:
'       in/out: L - the Left value
'               T - the Top value
'               W - the Width value
'               H - the Height value
'**********************************************************
Private Sub GetDimensions(L As Long, T As Long, W As Long, H As Long)

    ' Local variables declarations
    Dim DeltaX As Long
    Dim DeltaY As Long

    DeltaX = myPPT.ActivePresentation.PageSetup.SlideWidth -
WidthText.Text
    DeltaY = myPPT.ActivePresentation.PageSetup.SlideHeight -
HeightText.Text
    If DeltaX <= 0 Then
        L = 0
    Else
```

```vb
            L = DeltaX / 2
        End If
        If DeltaY <= 0 Then
            T = 0
        Else
            T = DeltaY / 2
        End If
        W = WidthText.Text
        H = HeightText.Text
        If W > 720 Then W = 720
        If H > 540 Then H = 540
End Sub
Private Sub lineInfo(searchStr As String, k As String, D As String, V
As String)
    Dim istart As Integer
    Dim istop As Integer
    istart = 1
    istop = 0
    istop = InStr(istart, searchStr, "=", vbTextCompare)
    ' Get the key string
    k = Mid(searchStr, istart, istop - 1)
    istart = istop + 1
    istop = InStr(istart, searchStr, "|", vbTextCompare)
    ' Get the directory string
    If istop > istart Then
        D = Mid(searchStr, istart, istop - istart)
        istart = istop + 1
        'Get the location string
        V = Mid(searchStr, istart)
    Else
        D = Mid(searchStr, istart)
        V = ""
    End If
End Sub
Private Sub downloadFile(URLStr As String, saveFile As String)
    Dim bData() As Byte        ' Data variable
    Dim intFile As Integer     ' FreeFile variable
    intFile = FreeFile()       ' Set intFile to an unused file.

    ' The result of the OpenURL method goes into the Byte
    ' array, and the Byte array is then saved to disk.
    bData() = Inet1.OpenURL(URLStr, icByteArray)
    Open saveFile For Binary Access Write As #intFile
    Put #intFile, , bData()
    Close #intFile
End Sub
Private Sub createFolder(path As String)
    Dim fs, f
    Set fs = CreateObject("Scripting.FileSystemObject")
    If Not fs.FolderExists(path) Then
        Set f = fs.createFolder(path)
    End If
    Set fs = Nothing
    Set f = Nothing
End Sub
Private Sub deleteFolder(path As String)
    Dim fs, f
```

```vb
        Set fs = CreateObject("Scripting.FileSystemObject")
        If fs.FolderExists(path) Then
            fs.deleteFolder path, True
        End If
        Set fs = Nothing
End Sub

Private Sub UserControl_ReadProperties(PropBag As PropertyBag)
        ServerURL = PropBag.ReadProperty("ServerURL", Nothing)
        'Debug.Print cfgFile & "      " & ServerURL
End Sub

Private Sub UserControl_Terminate()
        ' Delete the download folder
        deleteFolder downloadFolder
End Sub


'**********************************************************
'
' The Controller component uses this UpdateBrief event to
' notify the Continuous Brief wrapper (CBWrapper) for
' updating the brief.
' Event's parameters:
'       imageType: the type of images
'       imageLoc: the location where to find the images.
'
' The Glue component will raise the event to notify the
' Controller when it's done with storing data.
'
' The Monitor component will raise the event to notify the
' Controller when the new data come in.
' WithEvents causes the component(s) which raise the event(s)
' to run asynchronously.
' MonitorConnector component allows multiple connections to
' single Monitor object.
'
'**********************************************************
Event UpdateBrief(imageType As String)

Public WithEvents mGlue As Glue
Private WithEvents mMonitor As Monitor    ' Get Monitor events
Private mMonitorConnector As MonitorConnector

'**********************************************************
'
' Connect to the Monitor component
'
'**********************************************************
Private Sub Class_Initialize()

  Set mMonitorConnector = New MonitorConnector
  Set mMonitor = mMonitorConnector.Monitor

End Sub

'**********************************************************
```

117

```vb
'
' Receive the notification from the Monitor component
' The Controller passes the information to the Glue component
' for storing data to the database.
' Event's paramenter:
'       DataType: the data (images) type
'
'**********************************************************
Private Sub mMonitor_NewData(DataType As String)
    Set mGlue = New Glue
    Call mGlue.StoreData(DataType)
End Sub


'**********************************************************
'
' Receive the notification from the Glue component
' The Controller notifies the CBWrapper(s) and passes the
' information for the wrapper(s) to update the brief(s).
' Event's paramenter:
'       DataType: the data (images) type
'
'**********************************************************
Private Sub mGlue_GlueDone(DataType As String)       ' Asynchronous glue
component is done
    Set mGlue = Nothing      ' Free the Glue object

    ' Notify the CBWrapper for updating the brief
    RaiseEvent UpdateBrief(DataType)
End Sub

Public Sub GetImageInfo(ImageID As String, fileCounts As Integer, _
virtualDir As String, fileListName As String)

    Dim i As Integer
    For i = 1 To UBound(gCfgArray)
        If (StrComp(ImageID, gCfgArray(i).key, vbTextCompare) = 0) Then
            virtualDir = gCfgArray(i).vir_path
            fileListName = "CB_DATA.LST"
            Call makeFileList(fileCounts, gCfgArray(i).path, _
fileListName)
        End If
    Next
End Sub

Private Sub makeFileList(fileCounts As Integer, path As String, _
filename As String)
    Dim fs, f, fc, f1, i, j, a
    Dim myCount As Integer
    Dim listfileStr As String
    Dim myArray() As String

    ' Create a FileSystemObject for manipulating the file system.
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFolder(path)
    Set fc = f.Files
    myCount = fc.Count
    i = 1
```

118

```vb
    ' Store the name of the files to an array for sorting purpose
    ReDim myArray(1 To myCount)
    For Each f1 In fc
        myArray(i) = f1.Name
        i = i + 1
    Next

    ' Sort the array
    Call mMonitor.dhBubbleSort(myArray)
    listfileStr = path & "\" & "CB_listfile"
    createFolder listfileStr
    Set a = fs.CreateTextFile(listfileStr & "\" & filename, True)
    For j = (myCount - fileCounts + 1) To myCount
        a.WriteLine (myArray(j))
    Next
    a.Close
    ' Free up the objects, which are no longer be used.
    Set fs = Nothing
    Set f = Nothing
    Set fc = Nothing
    Set a = Nothing
End Sub
Private Sub createFolder(path As String)
    Dim fs, f
    Set fs = CreateObject("Scripting.FileSystemObject")
    If Not fs.FolderExists(path) Then
        Set f = fs.createFolder(path)
    End If
    Set fs = Nothing
    Set f = Nothing
End Sub
```

119

```
'#######################################################
'#  File: Glue.cls
'#  Date                Author              History            _
'#  5/31/2000           Tam Tran            Created.
'#######################################################

Option Explicit
'*********************************************************
'
' The Glue component uses this event to notify the
' Controller when done with its task.
' Event's parameter:
'        DataType: the data (images) type.
'
'*********************************************************
Event GlueDone(DataType As String)


'*********************************************************
'
' Notify the Controller when done storing data.
'
'*********************************************************
Public Sub StoreData(DataType As String)   ' Start glue task
    ' <Insert glue task here>
    ' ...
    RaiseEvent GlueDone(DataType)
End Sub
```

```vb
'#####################################################
'#   File: Monitor.cls
'#   Date                 Author          History
'#   5/31/2000            Tam Tran        Created.
'#####################################################
Option Explicit
'*****************************************************
' The VISStamDate, IRStampDate, and VAPORStampDate variables
' store the created date of the latest stored data.
'
' WithEvents causes the component(s) which raise the event(s)
' to run asynchronously.
' Event's parameter:
'        DataType: the data (images) type
'
' The Monitor component will raise the event to notify the
' Controller when the new data come in.
'*****************************************************
Private VISStampDate As Date
Private IRStampDate As Date
Private VAPORStampDate As Date

Private mTiming As Timing
Private WithEvents mClock As Timer

Event NewData(DataType As String)


' An array that holds the StampDate data type
'*****************************************************
'
' The tasks done when a new Monitor object is created.
'
'*****************************************************
Private Sub Class_Initialize()

    ' Start Monitor Timer and create instance of form
    Set mTiming = New Timing
    Load mTiming

    ' Connect timers' events to associated event procedures in Monitor
    Set mClock = mTiming.Clock

    ' Get the config information from the configuration file
    Call GetConfig
End Sub

'*****************************************************
'
' The tasks done when the Monitor object is terminated.
'
'*****************************************************
Private Sub Class_Terminate()    ' Terminate Monitor

    ' Free up the timer object.
    Set mClock = Nothing
```

121

```vb
    ' Unload and free up the form.
    Unload mTiming
    Set mTiming = Nothing
End Sub


'***********************************************************
'
' Process Timer Event.
' This timer event causes the Monitor to poll the storage
' directories for new data.
' The Monitor will raise the event(s) if it found a new data.
'
'***********************************************************
Private Sub mClock_Timer()
    Dim i As Integer
    For i = 1 To UBound(gCfgArray)
        If IsNewFile(gCfgArray(i).path, i) Then
            RaiseEvent NewData(gCfgArray(i).key)
        End If
    Next
End Sub


'***********************************************************
'
' The IsNewFile function is used to determine whether or
' not a new data exists.
' Paramenters:
'       in: StrDir - the directory where to check for
'                    new data.
'       in: StampDate - the created date of the latest
'                       data from the previous checked.
'       return: TRUE if there's new data, and FALSE otherwise.
'
'***********************************************************
Private Function IsNewFile(StrDir As String, arrayIndex As Integer) As
Boolean
    ' Local variables declarations.
    Dim fs, f, fc, f1, i
    Dim myStamp As Date
    Dim myArray() As String

    ' Create a FileSystemObject for manipulating the file system.
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFolder(StrDir)
    Set fc = f.Files
    i = 1

    ' Store the name of the files to an array for sorting purpose
    ReDim myArray(1 To fc.Count)
    For Each f1 In fc
        myArray(i) = f1.Name
        i = i + 1
    Next

    ' Sort the array
    Call dhBubbleSort(myArray)
```

122

```vba
    ' Check for new file based on the file's created date.
    myStamp = fs.GetFile(StrDir & "\" & myArray(fc.Count)).DateCreated
    If (DateDiff("s", gCfgArray(arrayIndex).stampdate, myStamp) <> 0)
Then
        gCfgArray(arrayIndex).stampdate = myStamp
        IsNewFile = True
    Else
        IsNewFile = False
    End If

    ' Free up the objects, which are no longer be used.
    Set fs = Nothing
    Set f = Nothing
    Set fc = Nothing
End Function

'**********************************************************
' Standard bubblesort.
' DON'T USE THIS unless you know the data is already
' almost sorted! It's incredibly slow for
' randomly sorted data.

' There are many variants on this algorithm.
' There may even be better ones than this.
' But it's not even going to win any
' speed prizes for random sorts.

' From "Visual Basic Language Developer's Handbook"
' by Ken Getz and Mike Gilbert
' Copyright 2000; Sybex, Inc. All rights reserved.

' In:
'   varItems:
'        Array of items to be sorted.
' Out:
'    VarItems will be sorted.
'**********************************************************
Public Sub dhBubbleSort(varItems As Variant)

    Dim blnSorted As Boolean
    Dim lngI As Long
    Dim lngJ As Long
    Dim lngItems As Long
    Dim varTemp As Variant
    Dim lngLBound As Long

    lngItems = UBound(varItems)
    lngLBound = LBound(varItems)

    ' Set lngI one lower than the lower bound.
    lngI = lngLBound - 1
    Do While (lngI < lngItems) And Not blnSorted
        blnSorted = True
        lngI = lngI + 1
        For lngJ = lngLBound To lngItems - lngI
            If varItems(lngJ) > varItems(lngJ + 1) Then
                varTemp = varItems(lngJ)
```

```vb
                    varItems(lngJ) = varItems(lngJ + 1)
                    varItems(lngJ + 1) = varTemp
                    blnSorted = False
                End If
            Next lngJ
        Loop
End Sub
'**********************************************************
'
' The lineInfo subroutine parses a line input from the
' configuration file (cbdata.cfg). It separates the
' image type value, and the location value from the input.
'
'**********************************************************
Private Sub lineInfo(searchStr As String, K As String, D As String, V _
As String)
    Dim istart As Integer
    Dim istop As Integer

    istart = 1
    istop = 0
    istop = InStr(istart, searchStr, "=", vbTextCompare)
    ' Get the key string
    K = Mid(searchStr, istart, istop - 1)
    istart = istop + 1
    istop = InStr(istart, searchStr, "|", vbTextCompare)
    ' Get the directory string
    If istop > istart Then
        D = Mid(searchStr, istart, istop - istart)
        istart = istop + 1
        'Get the location string
        V = Mid(searchStr, istart)
    Else
        D = Mid(searchStr, istart)
        V = ""
    End If
End Sub
'**********************************************************
'
' The GetDateArrayIndex function returns an index of the
' dateArray, where the specified image type (ID) is stored.
'
'**********************************************************
Public Function GetArrayIndex(key As String) As Integer
    Dim tmpInfo As cfgInfo
    Dim bFound As Boolean
    Dim i As Integer
    bFound = False
    i = 1
    Do While Not bFound
        tmpInfo = gCfgArray(i)
        If (StrComp(tmpInfo.key, key) = 0) Then
            GetArrayIndex = i
            bFound = True
        End If
        i = i + 1
    Loop
```

```
End Function
'*********************************************************
'
' The GetConfig subroutine reads information stored in
' the configuration file, and adds them to the link list.
'
'*********************************************************
Private Sub GetConfig()

    Dim cfgpath As String
    Dim inputStr As String
    Dim keyStr As String
    Dim dirStr As String
    Dim virDirStr As String
    Dim intFile As Integer
    Dim tmpInfo As cfgInfo

    ' Initialize the size the gCfgArray
    ReDim gCfgArray(0)
    ' Get the path for the configuration file
    cfgpath = Environ("CB_HOME") & "\cbdata.cfg"

    ' Store the configured info to the array
    intFile = FreeFile()
    Open cfgpath For Input As #intFile
    Do While Not EOF(intFile)
        Line Input #intFile, inputStr
        Call lineInfo(inputStr, keyStr, dirStr, virDirStr)
        With tmpInfo
            .key = keyStr
            .path = dirStr
            .vir_path = virDirStr
            .stampdate = -1      ' initialize the date to before Dec.
30, 1899
        End With
        ReDim Preserve gCfgArray(UBound(gCfgArray) + 1)
        gCfgArray(UBound(gCfgArray)) = tmpInfo
    Loop
    Close #intFile
End Sub
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1) [Berson] *Client/Server Architecture*, Alex Berson, McGraw-Hill, 1996.

2) [Bloor] *The Object Management Architecture Guide*. Robin Bloor, Butler Bloor Ltd., Challenge House, April 1992.

3) [Carr] *CORBA and DCOM: How Each Works*, David F. Carr, http://internetworld.com/print/1997/03/24/software/cobra.html

4) [DISA] *Recommendations for Using DCE, DCOM, and CORBA Middleware*, DII COE Distributed Application Series, David Diskin, DISA/MITRE Corporation, April 1998.

5) [Gee] *An Architectural Framework for Integrating COTS/GOTS/Legacy Systems*, Karen Gee, Master's Thesis, Naval Postgraduate School, Monterey, CA, June 2000.

6) [Microsoft] *Transactional Component Services*, Microsoft Corporation, http://www.microsoft.com/com/mts-f.htm, 1998.

7) [NPS] *Interoperability Technology Assessment for Joint C4ISR Systems*, Technical Report Number NPSCS-00-001, Naval Postgraduate School, 1999.

8) [OMG] *CORBA Overview*, & *The OMA Reference Model*, OMG, http://www.infosys.tuwien.ac.at/Research/Corba/OMG/arch2.html, 1995.

9) [Orfali] *Client/Server Survival Guide*, Robert Orfali, Dan Harkey, Jeri Edwards, Wiley Computer Publishing, 1999.

10) [Quoin] *What is Java*, Technical Report, QUOIN, 1998.

11) [Redmond] *DCOM: Microsoft Distributed Component Object Model*, Frank E. Redmond III, IDG Books, 1997.

12) [Rosenberger] *Teach Yourself CORBA in 14 Days*, Jeremy Rosenberger, SAMS Publishing, 1996-1999.

13) [Software AG] *XML- The Benefits*, Software AG,
    http://www.softwareag.com/xml/about/xml_ben.htm, 2000.

14) [SPAWAR] *Performance Specification (PS) for the
    Tactical Environmental Support System/Next Century TESS
    (NC)(AN-UMK-3)(NITES version I and II)*, SPAWARSYSCOM
    METOC PMW-185, 1998.

15) [SPAWAR PMW-185] *Programmer's Guide for Data Retrieval
    via METCAST*, SPAWARSYSCOM PMW-185, 1999.

16) [Tran] *Interoperability and Security Support for
    Heterogeneous COTS/GOTS/Legacy Component-Based
    Architecture*, Tam Tran and James Allen, Master's
    Thesis, Naval Postgraduate School, Monterey, CA, June
    2000.

17) [Wallace] *Learn Microsoft Transaction Server
    Development Using Visual C++ 6.0*, Nathan Wallace,
    Wordware Publishing, 1999.

18) [Yang] *CORBA: A Platform for Distributed Object
    Computing*, Technical Report, Zhonghua Yang and Keith
    Duddy, 1997.

LIST OF ACRONYMS

ADO	ActiveX Data Object
AO	Applications Objects
API	Application Program Interface
BOA	Basic Object Adapter
CF	Common Facilities
CLI	Call Level Interface
CLSID	Class Identifier
COM	Component Object Model
CORBA	Common Object Request Broker
COTS	Commercial-Off-The-Shelf
ORB	Object Request Broker
C4ISR	Command, Control, Communications, Computers, Information, Surveillance and Reconnaissance
CCTV	Closed-Circuit Television
CDR	Common Data Representation (CDR)
DCE	Distributed Computing Environment
DCOM	Distributed Component Object Model
DDCF	Distributed Document Component Facility
DISA	Defense Information System Agency
DLL	Dynamically Linked Library
DoD	Department of Defense
DoN	Department of the Navy
DTD	Document Type Definition
EJB	Enterprise JavaBeans
EXE	Executable
GIOP	Generic InterORB Protocol
GOTS	Government-Of-The-Self
GUID	Globally Unique Identifiers
IDL	Interface Definition Language
IIOP	Internet Inter-ORB Protocol
IOR	Interoperable Object Reference
IPC	Interprocess Communications
JDBC	Java Database Connectivity
JVL	Java Virtual Language
LAN	Local Area Network
LRPC	Lightweight Remote Procedure Call
METOC	Meteorological and Oceanographic
MTS	Microsoft Transaction Server
NITES I	Navy Integrated Tactical Environmental Support System I
NT	New Technology
ODBC	Open Database Connectivity
OLE	Object Linking and Embedding
OMA	Object Management Architecture
OMF	Observation Definition Format

OMG	Object Management Group
ORPC	Object Remote Procedure Call
OS	Object Services
OTMs	Object Transaction Monitors
PI	Processing instructions
RDBMS	Relational Database Management System
RPC	Remote Procedure Call
SPAWAR	Space and Naval Warfare
SQL	Structured Query Language
SRS	Software Requirements Specifications
TCP/IP	/Internet Protocol
UDP	User Datagram Protocol
VFTL	Virtual Function Table

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center   2
    8725 John J. Kingman Rd., STE 0944
    Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library, Code 52   2
    Naval Postgraduate School
    Monterey, CA 93943-5100

3. Dr. Dan Boger, Code CS   1
    Computer Science Department
    Naval Postgraduate School
    Monterey, CA 93943-5118

4. Dr. Luqi, CS/Lq   1
    Computer Science Department
    Naval Postgraduate School
    Monterey, CA 93943-5118

5. Dr. Mantak Shing, CS/Sh   1
    Computer Science Department
    Naval Postgraduate School
    Monterey, CA 93943-5100

6. Thomas Nguyen   1
    SPAWARSYSCEN Code D871
    53560 Hull Street
    San Diego, CA 92152-5001